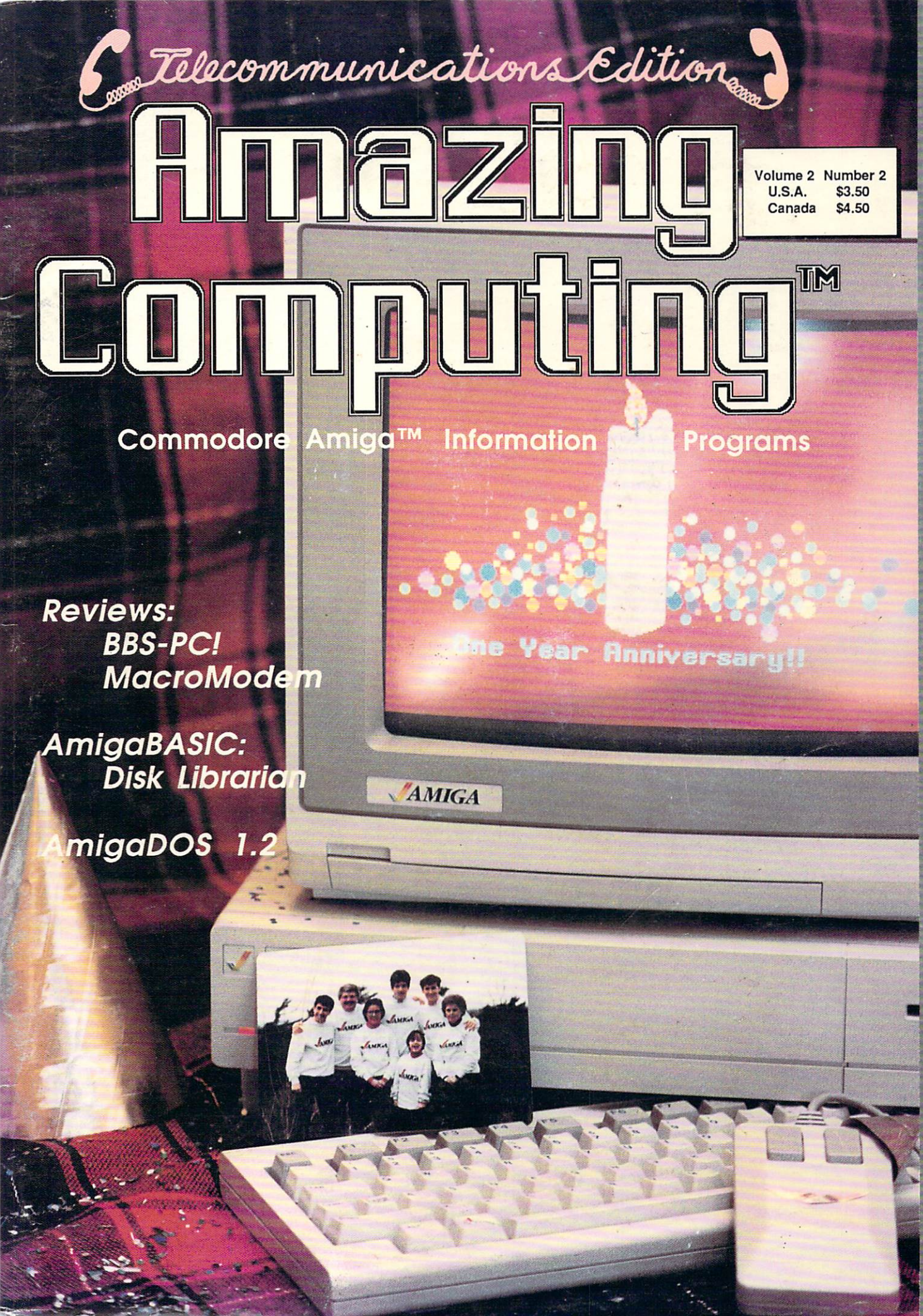
Telecommunications Edition

Amazing Computing™

Volume 2 Number 2
U.S.A. $3.50
Canada $4.50

Commodore Amiga™ Information Programs



Reviews:
BBS-PC!
MacroModem

AmigaBASIC:
Disk Librarian

AmigaDOS 1.2

BUILD YOUR OWN MIDI INTERFACE

Superbase™ PERSONAL
RELATIONAL DATABASE SYSTEM

SUPERBASE IS NOW AVAILABLE FOR THE AMIGA

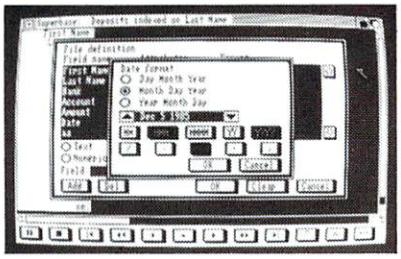
The enormously popular and proven database system, Superbase is now available for the Amiga computer. We completely rewrote Superbase to take full advantage of all the power available on the amazing Amiga. This is not a conversion, but an entirely new Amiga program!

TOTAL SOLUTIONS

Superbase provides the total information management solution. It is a true productivity program for the Amiga computer. You can finally use a serious database with a serious machine. It's easy to keep track of inventories for your business whether you're working with parts inventory or real estate listings. Superbase is perfect for church membership rolls, patient files, personnel schedules or any place you need to manage and control large amounts of important information.

Access the power of the first **true** relational database management system with Superbase Personal Relational Database System. It will turn your Amiga into a truly productive tool, with virtually limitless capacities. Imagine being able to have an unlimited number of files open at any time. You can even have each file indexed with up to 999 key fields to search and sort.

EASY TO SET UP



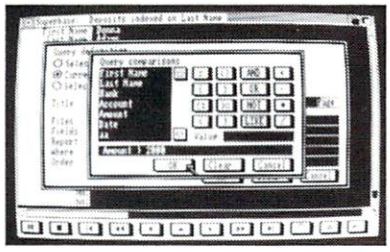
Superbase utilizes the latest ideas in easy-to-use mouse and windowing technology. There are pull-down help menus to ease you through problems that may occur during database creation. Superbase is completely menu driven and takes advantage of the point-to-click features possible with the Amiga mouse.

Create a database in minutes using the easy to understand menu selections and control panels. Type in field names, add details like length or data style and you are quickly ready to input your data. Unlike other databases, you can alter your formats at anytime, **without disturbing** the data already in existing files. Using Superbase's Enhanced BASIC, your database can be totally customized to virtually any application.

IT'S EASY TO MANAGE YOUR DATA

Display your data in the format you choose. Either page by page or just as it appears in the record format. You choose how to view the data you need. There is practically no limit to the number of fields in a record, you have complete control over what is displayed on screen or printed in custom reports.

Decide on the fields and on the sequence, then use the VCR style controls to view your data -- Get the first record, then fast forward, pause, continue or stop -- it's as easy as playing a video tape!



WORKING POWER

Superbase makes it easy to define reports or generate relational queries across multiple files, with multiple sort levels if you need them. Import data from other databases or applications. Export data to your favorite word processor, or join several files to form a new database,

The advanced B+ tree file structure and disk buffering means high performance -- Superbase reads a typical name and address record in an incredible three hundredths of a second!

THE VIDEO DATABASE

Superbase includes an amazing array of data types in your record format, including the names of pictures or digitized images stored on disk. Read the words, then look at digitized pictures you have already stored on disk. Your data records can "point" to images to recall them for viewing!

You can even link multiple images to a single database record to run automatic slide shows. It's all easily done using the VCR style commands that you control. Revise, update or review your illustrated database in any desired arrangement. You have total control! Superbase is the total software solution for people who must manage information.

THE BEST HAS ARRIVED!

Finally, a program that utilizes ALL the power and functions of the Amiga computer. Superbase brings to the Amiga the business solutions you have been waiting for.

The power of Superbase is also available for the Commodore 64/128 and the Apple IIe/IIc.

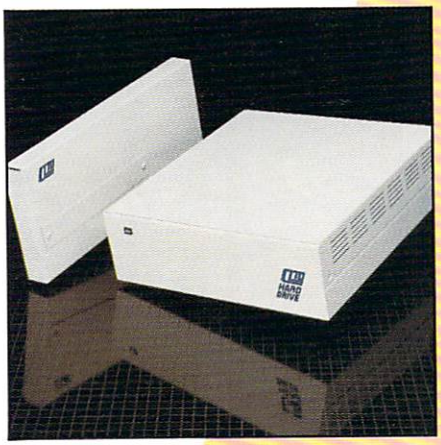
WHEN QUALITY COUNTS!

PROGRESSIVE PERIPHERALS & SOFTWARE

464 KALAMATH STREET
DENVER, COLORADO 80204
303-825-4144
TELEX: 888837

20-Meg SCSI Hard Drive $999.95

- Full AutoConfig
- Full Pass-Through out of Amiga expansion port
- Controller Supports 7 additional devices
- Internal Power Supply
- Faster than any comparably-priced drive



Create your own hard drive system

Buy Components Separately

Hard Drive only $799.95

SCSI Controller $299.95

JetSet

Amiga Laser Printing Software $69.95

- Works with Hewlett Packard LaserJet™ or compatible laser printer
- Hundreds of Fonts available (starter typeface included)
- Works with Textcraft™ & Scribble™

JetSet Fonts $49.95 to $99.95

Complete Typeface in each package (e.g. italic, bold italic, bold, demi-bold, regular in variety of sizes)

Selection Includes...Times ■ Triumvirate ITC Souvenir ■ Old English ■ Unical Commercial Script ■ Dom Casual ITC Benguiat Bold ■ Broadway Globe Gothic Outline ■ Borders Symbols ■ ITC Dingbats ITC Souvenir Greek/Math ITC Times Greek/Math. And many, many more.

aMEGA Board $549.95

Million Bytes of RAM

- Full AutoConfig Compatibility
- Works with all popular Amiga software
- Pass-Through for future expansion
- 6-Month parts & labor warranty

Available NOW at Amiga Dealers!

C Ltd. / 723 East Skinner / Wichita, KS 67211 / (316) 267-6321

ALIEN FIRES

2199 AD

TIME LORDS:
An immortal race of beings responsible for protecting the temporal balance of the universe from chaos and destruction.

Alien Fires uses every ounce of the Amiga's power to create a state-of-the-art visual and aural sensation, with colourful full-screen 3 dimensional computer graphics, digitized music and sound effects. It is an experience of incredible realism and sophistication that takes you through time and space to a new dimension in entertainment.

Play the part of a Time Lord in this imaginative new series of role-playing adventures. Alien Fires — Part I, 2199 AD, is now ready to take you into the distant future, in search of a great man and an awesome device capable of twisting the very fabric of time itself.

Jagware Inc.

For the dealer nearest you, call toll-free:
1-800-267-1904
Dealer inquiries invited

Amazing Dealers

The following are Amazing Dealers, dedicated to supporting the Commodore-Amiga™. They carry Amazing Computing™, your resource for information on the Amiga™.

If you are not an Amazing Dealer, but would like to become one, contact:

PiM Publications, Inc.
P.O.Box 869
Fall River, MA. 02722
1-617-678-4200

Alabama
Alabama Book Store — Tuscaloosa
Abax Data Systems — Huntsville
Command Computers — Birmingham
Electronics Boutique — Hoover
Madison Books & Computers — Madison
Mid Tel Communications — Mobile
Mel's Photo & Computer Shop — Montgomery
Universal Computer System — Mobile

Arkansas
SIS Inc. — Mt. Home
The Micro Shop — Little Rock

Arizona
Computer Library, Inc. — Tempe
Computer West — Phoenix
Copperstate Business Systems — Phoenix
Copperstate Computers — Tucson
Hacke's — Tempe
North American Digital — Tuscon
Tronixtools — Tuscon

California
Bits n Pieces Comp Prod. — Anahien
Brown Knows Computing — Redlands
Byte Factory Computers — Selings
C.F.T. — Clairemont
Century Computer Systems — La Habra
Compare Inc. — Ridgecrest
Computer Attic — Palo Alto
Computer Junction — Santa Ana
Computer Literacy Bookshop — Sunnyvale
Computer Nook, Inc. — San Bernadino
Computer Technology — Pomona
Computer Time — Citrus Hts
ComputerLand Fresmont — Fresmont
ComputerLand of Stockton — Stockton
Dataphile — Watsonville
Electronics Boutique — West Los Angeles
Empire Computers — Santa Rosa
Gary Bell — Santee
Hanging Way News — Stockton
Home Computing Center — Concord
Home Computing Center — Mill Valley
Home Computing Center — San Leandro
Home Computing Center — San Bruno
HT Electronics — Sunnyvale
K J Computers — Granada Hills
Levity — North Hollywood
Micro Flight — San Jacinto
Nonagon Software — El Cerrito
Op Amp Tech Books — Los Angeles
Paul Diaz — San Rafael
R&R Software — Chula Vista
Ridgecrest Computer Ctr. Inc. — Ridgecrest
S.O.S. Computers — Los Angeles
Software First — Santa Rosa
Software Plus — Citrus Heights
The Computer Room — Scotts Valley
The Floppy Disk Inc. — Downey
Winner's Circle — Berkley

Colorado
Citadel Computers — Colorado Springs
Computer Discount — Denver
Computer Room — Aurora
Micro Computer World — Fort Collins
Micro World — Lakewood
Sun Country Computers — Colorado Springs
Whole Life Distributors — Englewood
Whole Life Distributors — Englewood

Connecticut
Electronics Boutique — Danbury
Mnemonics Inc. — Norwalk
Personal Computer center — Norwich
Software City — Orange
Software Kingdom — East Windsor
Spectrum Computers — New Haven

Delaware
Castle Video — Newark

Florida
A A Computers — Jacksonville
Book Mania — Orlando
Commodore Computer Shop — Ocala
Computer Bar — Pennsacola
Computer Base Enterprises Inc. — Tampa
Computer Corner — Tampa
Computer Image — Miami
Computer Terminal Inc. — Sebring
Computers 4 Rent — W. Palm Beach
Computers Plus — South Daytona
Cotronics Inc. — Stewart
Education Computers Etc. — Talahasse
Electronics Boutique — Boynton Beach
Electronics Boutique — Tallahassee
Electronics Boutique — Miami
Electronics Boutique — Orlando
Family Computers — Sarasota
Florida Book Store — Gainesville
Games 'N' Gadgets — Orange Park
Games 'N' Gadgets — Jacksonville
Games 'N' Gadgets — Tallahassee
Games 'N' Gadgets — Clearwater
Gulf Coast Computer Exchange — Panama City
K&L Computers — E. Palatka
Megasport Computer Center — South Venice
Micro Computer Center — Largo
Micro's Etc. — Altamonte Springs
MSI Business System — Brooksville
New Age Electronics — St. Petersburg
Rainbow Comptr Center — Orlando
Random Access Computers Inc. — Ft. Walton Beach
Softerals Computer Center Inc. — Holy Hil
The Open Door — Cocoa
The PC Collection — Coral Gables

Georgia
Electronics Boutique — Atlanta
Electronics Boutique — Atlanta
Electronics Boutique — Kennesaw
Future System — Augusta
Games 'N' Gadgets — Atlanta
Games 'N' Gadgets — Atlanta
Michael Dannellan — Decatur
Software South Inc. — Savannah
The 64 Store — Atlanta

Hawaii
Astro Computers & Electronics — Aiea
Computer House Inc. — Honolulu

Idaho
A.B.I. Computer & Video — Boise
ABI Computer & Video — Nampa

Illinois
A Bitokster — Rockford
BA Business Computers — Batavia
Computer Resolution — Champaign
Computerland O'Hare — Park Ridge
ComputerLand of Niles — Niles
I.C.V. Data Center Inc. — Mundelein
Illini Microcomputers Inc. — Naperville
Interstate Computer Service — LeMont
Software Or Systems — Fairview Hts.
The Memory Expansion — Normal
Unique Computer — Sterling

Indiana
Burket Computer Ctr. — South Bend
Bytes Computer — Ft. Wayne
Bytex of Angola — Angola
Computer Corner — Marion
Computer Corner — Fort Wayne
Computer People Inc. — Michigan City
Greg Chaney — Greenfield
Micro Computer Centers Inc. — Indianapolis
Spectra Computer Sys. Inc. — Evansville
Vons Computers — West Lafayette

Iowa
Century Systems — Des Moine
Gramat Computers — Des Moine
Micro Computer Applications LTD — Marshalltown

Kansas
Data Bank Corp. — Wichita
J & L Electronics — Liberal
MidKansas Computers — Newton
Midwest Computers — Manhattan
Thorobred Computers — Topeka

Kentucky
Computer Dimensions Inc. — Lexington
Microage Computer Store — Lexington
Reality World Computers — Lexington
Software Solutions — Baton Rouge
The Software Source — Louisville

Louisiana
Computer Time — Shreveport
Electronics Boutique — Kenner
Electronics Boutique — Monroe
Electronics Boutique — Laurenceville
Modern Business Machines — Metairie
Software Center International — Metairie
Software Source — Lafayette
The Computer Clinic — Lafayette

Maryland
Computer Crafters — Wheaton
ComputerWorld — Ocean City
Compuvision Vision Center — Columbia
Electronics Boutique — Baltimore
Electronics Boutique — Baltimore
Electronics Boutique — Anne Arundel County
Electronics Boutique — Owings Mills
Games 'N' Gadgets — Baltimore
Games 'N' Gadgets — Baltimore
Games 'N' Gadgets — Columbia
Micro Computer Center — Baltimore
Software Advantage — Rockville
Software N Things — College Park
The Logical Choice Inc. — Baltimore
Waldorf Computer — Waldorf

Massachusetts
Club Computer — Cambridge
E.U. Wurlitzer Co. — Boston
Electronics Boutique — Holyoke
General Computer Store — Framingham
HCS Computer Center — Marshfield
LCA Video & Computer Ctr. — Norwood
Memory Location — Wellsley
Metro Computer — Brookline
OmniTek Computers — Tewksbury
Pioneer Valley Data Equipment — Amherst
Tech Computer Store — Cambridge
The Bit Bucket — West Newton
The Software Haus — Quincy
Tycom Inc. — Pittsfield

Maine
Computer Barn — Damariscotta
Valley Computers — Auburn

Michigan
Basic Business Control Sys. — Traverse City
Basic Computer Center — Grand Rapids
Bits-Bytes-Nibbles — Petowskey
Canton Computer — Canton
CompuSoft — Kalamazoo
Direct Access — Lansing
Edmunton Computer Center — Pontiac
Galaxy Computers — Allen Park
Global Computer Center — Dearborn
Lovawes Computer Inc. — Adrian
Michigan Software — Novi
Midland Computer Shop — Midland
Pro-Video — Okemos
Professional Computer Systems — St. Joseph
Programs Unlimited — Grand Rapids
Retail Computer Center Inc. — Farmington Hills
Rossville Computer Store — Warren
Slip Disk — Madison Heights
Software Plus — Grand Rapids
Specialists Computer — Livonia
State Street Computers — Ann Arbor
T.J. Enterprises — Clinton
The Software House — Kalamazoo
Ye Olde Computer Shoppe — Ypsilanti

Minnesota
Computers, etc. — Eagan
Computer Outfitters — Roseville
Computer Place — Minneapolis
Computer Specialties Plus — Morriscllo
JMH Software — Maple Grove
MCD of Hobbing Inc. — Hibbing
OnLine Computers — Minnetonka
Specialists IN — Hopkins

Mississippi
Electronics Boutique — Ridgeland
Enterprises Unlimited — Jackson
National Computer Center — Ocean Springs

Missouri
Associated Computer Services — Springfield
Brands Mart Computers — Kansas City
Data Byte Computer Co. — St. Louis
Instant Replay Ltd. — Chesterfield
Systems Plus — Independence
Systems Plus of St. Louis — St. Louis

Montana
Appelgren Computer Systems — Great Falls

Nebraska
BULLS-EYE — Lincoln
My Com Inc. — Plattsmouth
Doubls E Electronics — Omaha

New Hampshire
Computer Mart of New Hampshire — Nashua
Diversified Computers — Keene
Electronics Boutique — Nashua
Ormisk — Salem

New Jersey
Electronics Boutique — Voorhees
Electronics Boutique — Wayne
Electronics Boutique — Rockaway
Electronics Boutique — Cherry Hil
Electronics Boutique — Woodbridge
Family Computer Centres — Fairfield
Family Computer Centres — South Orange
Family Computing Centres — Ridgewood
Games 'N' Gadgets — Burlington
Games 'N' Gadgets — Livingston
Morris County Stationers — Chester

New Mexico
Academy Computers — Albuquerque
CVM Systems — Santa Teresa
New Horizon Computer System — Alamogordo
Page One Newstand — Albuquerque
Santa Fe — Santa Fe
Technological Concepts — Las Cruces

New York
Amuse — New York
Amuse- Hudson Valley — Wappingers Falls
Amiga Business Computer — North Port
Byte Shop — Merrick
Castle Computer — Latham
CIA Software Center — Flushing
Compurite Center — White Plains
Computer Outlet — Jamestown
Computers Etc. — Syracuse
Electronics Boutique — Lake Grove
Electronics Boutique — Huntington Station
Electronics Boutique — Albany
Electronics Boutique — Valley Stream
Games 'N' Gadgets — Nanuet
Games 'N' Gadgets — Garden City
Leigh's Computers — New York
Ray Supply Inc. — Glen Falls
Software Centre International — Forest Hils
Software & Such — Scotia
Software City — Forest Hils
Software Supermarket — Kenmore
Star Tech Systems — Massena
Tedrow Business Products — Rochester
The Computer Cellar — Albany
The Working Computer — Stoney Brook
Video Computer Bin — Brooklyn
Video Computer Center — Rome
Video Connection — Schenectady
Village Computer Inc. — New York
World Computers Inc. — Hickaville

North Carolina
Digitz — Raleigh
Games 'N' Gadgets — Raleigh
Software City — Raleigh
Software City — Charlotte
TDS Computers — Durham
TDS Computers — Durham
Triad Computers — Greensboro

North Dakota
Computer Associates — Fargo
G. F. Computers — Grand Forks
The Computer Store — Minot

Nevada
Century 23 Inc. — Las Vegas
Computer World — Las Vegas
ComputerLand of Carson — Carson City

Ohio
Computer Network — Newark
Computer Site — Cleveland
Computers Plus of Ohio — Upper Sandusky
Earthrise Micro Systems — Columbus
Electronics Boutique — Columbus
Fairborn Home Computer — Fairborn
Games 'N' Gadgets — North Randall
Lakes Consumer Electronics — Akron
Main Office Supply — Coshocton
Micro Center — Columbus
Microwave Magic — Fairfield
North Coast Programming — Wiloughby
Program Store — Columbus
Quality Computer Appliances — Toledo
Quality Computer Appliances — Toledo
Sarton Computer Center — Sandusky
Software Centre International — North Olmsted
Software and More — Cincinnati
Software Connection — Mayfield Hts.

Oklahoma
Colonial Video & Computer — Bartlesville
Micro Link Computer Store — Tulsa
P C Tech — Tulsa
Second Hand Software — Oklahoma City
Video-Comp Inc. — Lawton

Oregon
BTE Telephone & Computer — Klamath Falls
Clackamas Computers — Beaverton
Clackamas Computers — Clackamas
Computron Business Systems — Portland
IB Computers — Portland
Software Express — Eugene
University of Oregon Bookstore — Eugene

Pennsylvania
Alpha Omega EBE — Warren
Basic Computer Systems — Hermitage
Circle Computer — Ephrata
Data Softique Computerware — Pittsburg
Downington Computer Center — Downington
East Coast Software — Hershey
Electronics Boutique — North Wales
Electronics Boutique — Langehorne
Electronics Boutique — Exton
Electronics Boutique — Norristown
Electronics Boutique — Pittsburgh
Electronics Boutique — King of Prussia
Electronics Boutique — Whitehall
Games 'N' Gadgets — Lancaster
Games 'N' Gadgets — King of Prussia
Games 'N' Gadgets — Media
Games 'N' Gadgets — West Mifflin
Pittsburgh Computer Store — Pittsburgh
The Electronic Boutique — Norristown
Triangle Computers — Indiana

Rhode Island
Micro Limits — Smithfield
International Computer Services — Johnston
Micro Limits — Warwick

South Carolina
Avcom International Inc. — West Columbia
Electronics Boutique — Greenville
Horizon — Greenville
Micro Computer Depot — Sumter
Software Solutions — Westwood Plaza
Watson's Jewlery — Chesterfield

South Dakota
Computer Business Center — Rapid City

Tennessee
ComputerLab of Memphis — Memphis
Games 'N' Gadgets — Memphis
Software First — Nashvile

Texas
A. B. Computer — Round Rock
Byte Shop — Austin
C.E.M. Corporation — Brazoria
Colonial Video — Houston
Computer Age — Houston
Computer Age — Dallas
Computer Concepts — Waco
Computer Magic — Austin
Computer Revelations — Houston
Computer Time Inc. — Marshall
DataPro — Wichita Falls
FNS Computer Services — Texarkana
Metropolitan Computer — Richardson
Micro Search — Huston
Regency Educational Systems — Dallas
Softex — Watauga
Software Centre — Dallas
Software Library — Wichita Falls
Software Terminal — Fort Worth
The Computer Experience — San Antonio
The Computer Stop — Abalean

Utah
Computers Plus — Murray

Virginia
Colony Ltd. — Hampton
Diskovery — Falls Church
Family Computer Center — Fairfax
Games 'N' Gadgets — Hampton
Games 'N' Gadgets — Virginia Beach
Games 'N' Gadgets — Virginia Beach
Micro Enhancement — Virginia Beach
Software Central — Norfolk
Software City — Richmond
Virginia Micro Systems — Woodbridge

Washington
A.P.P.L.E. CO-OP — Renton
Apple Valley Computers — E. Wenatchee
Bull Dog News — Seattle
Com-Soft Computers — Everett
Computer & Movie Magic — Yakima
Computers + Inc. — Tacoma
Family Computers — Lynwood
Micro Age Computer Store — Bellview
Nibble Site — Tacoma
Omni International — Seattle
PC Tech — Kirkland
Programs Plus — Seattle
Take-A-Byte — Kennewick

Wisconsin
Calcetron Computers — Racine
Computer Software Center — Milwaukee
Mom's Computers — Stevensport
Software Shoppe — Fond Du Lac
TMW Software — Greenbay
TMW Software Inc. — Wausau

Canada
Compute or Play — Alberta
Computerworks — Alberta
Computer Shop — Alberta
Data Pro — Alberta
Computer Shop of Calgary — Alberta
The Software House — Alberta
Hindsin Computing — Alberta
Red Deer Micro Systems — Alberta
Memory Land — Alberta
Computer King — British Columbia
Cord — British Columbia
Nutec — British Columbia
Sprita — British Columbia
Strider — British Columbia
Super Software — British Columbia
Canadian Computer Era — Manitoba
Micromart — Manitoba
ComputerWorld — Manitoba
J.W. Allen — Manitoba
Kobetak — Manitoba
Minicomp — Manitoba
Arkon — Ontario
Chatam Computer — Ontario
Computerland — Ontario
Comspec — Ontario
Electronics 2001 — Ontario
Hastings Data — Ontario
The Computer & You — Ontario
Thornhil — Ontario
Software Co. — Quebec
Detronix Solare — Quebec
Marocora — Quebec
Info Mercier — Quebec
Laval Micro — Quebec
Vision Tronique — Quebec
Distribution Visipro — Quebec

Distributed in Canada by
Avita Software Distributors — Concord, Ontario
Compuit Distributors — British Columbia
Phase 4 Dist. Inc. — Calgary, Alberta

Argentina
Logic Computer — Buenos Aires

Australia
High Technology Computer Supplies — Brighton, Melborn

Japan
Pinespple 6502 — Tokyo, 111

Saudia Arabia
Abdulla Fouad & Sons — Daman

Amazing Computing™

Publisher: Joyce Hicks
Circulation Manager: Doris Gamble
Assistant to the Publisher:
Robert James Hicks
Traffic Manager: Robert Gamble

Managing Editor: Don Hicks
Assistant Editor: Ernest P. Viveiros Jr.
Hardware Editor: Ernest P. Viveiros
Amicus & Technical Editor: John Foust
Music Editor: Richard Rae
Art Director: Keith Conforti
Assistant Advertising Manager:
John David Fastino
Production Manager: Mark Thibault

Amazing Authors
Ervin Bobo
Bryan Catley
John Foust
Don Hicks
Kelly Kauffman
Perry Kivolowitz
George Musser Jr.
Steven Pietrowicz
Rick Wirch
&
The Bandito

Special Thanks to:
Robert H. Bergwall
RESCO, Inc.
E.P.V. Consulting
New England Technical Services
Software Supermarket

Advertising Sales
&
Editorial

1-617-678-4200

Amazing Computing™ (ISSN 0886-9480) is published by PiM Publications, Inc. P.O. Box 869, Fall River, MA. 02722. Subscriptions: in the U.S. 12 Issues for $24.00; Canada and Mexico, $30.00; Overseas, $35.00. Printed in the U.S.A.

First Class or Air Mail rates available upon request.

PiM Publications, Inc. maintains the right to refuse any advertising

Amazing Computing™

Amazing Contents

Volume 2 Number 2

Amazing Features...

Amazing Reviews...

Amazing Columns...

Amazing Departments...

From The Editor:

Looking Back
This is the first anniversary of Amazing Computing™. One year ago, a few people (very few) packed the hundred or so orders for a magazine no one had ever seen. The packing was completed in our kitchen, and almost all the orders were ready by the time the UPS driver arrived. Since then, it has been both a continual race and pleasure to provide information on the Commodore Amiga.

As I look through the ten issues we have completed in the past year, I am proud of what we have accomplished, of what we have covered, and of what the Amiga has slowly but surely been able to do.

The stack of magazines is impressive. I had not considered just how much we had done. I will not trouble you with the list of articles (there is a condensed list on page 48), except to say that they were the work of many Amiga enthusiasts across the continent. Writers, some having never seen the magazine, provided articles in the hope that the Amiga would be realized for the adaptable machine it is.

The list of advertisers has grown from those early issues, not so much from our almost non-existent advertising sales staff, but from the growth and acceptance of the Amiga. Software and hardware producers are releasing more products each month with a wider variety of applications and features. They are taking a greater advantage of the capabilities of the Amiga

This year did have its problems. Early Amiga owners have faced an uphill battle. Commodore suffered the blind arrogance of almost every major trade publication and national magazine. We have seen Commodore exceed in improving their profitability (although, several good friends were laid off). We have watched as Commodore slowly implemented new products, revised old ones, announced new items and then delayed shipment. It has been both a good and a hard year for Commodore who was forced to make many difficult decisions. However, the truth is they are still here and even stronger than before.

It is interesting to note how many magazines and dealers use the Amiga's features as a comparison for other new machines. Apple's II GS is continually compared with the Commodore Amiga and the Atari ST. It is interesting to watch one of this industries greatest innovators playing "technology tag" with the Amiga.

Looking Ahead
With the Consumer Electronics Show approaching (by the time you see this, the show will have passed), the rumors have been flying as to what Commodore will show. Although no one will say what Commodore has planned, no one doubts Commodore will continue to develop and support the Amiga.

With new technology in graphics and video, Amiga has opened new markets for microcomputers in the new market of "Desktop Video Production". Several companies are working on their own versions of a live frame grabber and genlock devices. With software products such as Video Construction Set, Aegis Images, and their successors, a brand new capability has been handed to the general public.

As most of us are aware, the road to a computer's market acceptance must, at one time, run through the corporate board room. With presentation graphics provided by these new products, the old adage "I want to see it in Black and White" will become, "Give me a full video presentation for tommorow's meeting."

And the products keep arriving. This month, Amazing Computing has received new versions of existing software and some translations from other machines. DeluxePaint II has been released from Electronic Arts as well as an Amiga version of their DeluxeMusic Construction Set. Ultima III for the Amiga is now delivering. In short, we have seen products from Bible software to Strip Poker programs.

The Amiga holds more "promise" than ever before for new business. If you are a developer, the Amiga is well worth your consideration. Although the market is growing rapidly and Commodore is predicting a 200% increase in users as a conservative estimate, there is still room for innovative companies to take advantage of the Amiga.

THANK YOU!
There is no way that we could say thank you loud enough. We would not be here without the support and praise from you, our readers. Our writers have come from your ranks, our new advertisers have come from you, and our hopes and inspiration always come from you. Thank you!

Don Hicks
Managing Editor

Special Thanks to T's Me (see their ad on page 52) for their help in supplying the Sweatshirts used by our team on the front cover. The sweatshirts we received were not only good looking and comfortable, but they hold up through the wash. It's one of those computer peripherals you don't need, but are great to own!

Software designed for AMIGA.

Lattice® C Compiler — *$225.00*

New version 3.1 of the AMIGA DOS C Compiler replaces version 3.03. Major enhancements include the addition of: TMU, an assembler, a faster linker and version 3 MS-DOS.

With more than 30,000 users worldwide, Lattice C Compilers set the industry standard for MS-DOS software development. Lattice C gives you all you need for development of programs on the AMIGA. Lattice C is a full implementation of Kernighan and Ritchie with the ANSI C extensions and many additional features.

Professional Lattice® C Compiler — *$375.00*

A new product called the Professional Lattice C Compiler is now available. It includes the C Compiler package (complete with TMU), plus LMK, LSE and the Metascope Debugger.

AMIGA C Cross Compiler — *$500.00*

Allows AMIGA development on your MS-DOS system. Price includes the Professional Lattice C Compiler described above.

Lattice Screen Editor (LSE™) — *$100.00*

Designed as a programmer's editor, *Lattice Screen Editor (LSE)* is fast, flexible and easy to learn. *LSE's* multi-window environment provides all the editor functions you need including block moves, pattern searches and "cut and paste." In addition, *LSE* offers special features for programmers such as an error tracking mode and three Assembly Language input modes. You can also create macros or customize keystrokes, menus, and prompts to your style and preferences.

Lattice dBC III™ Library — *$150.00*

The *dBC III library* lets you create, access and update files that are compatible with Ashton-Tate's dBASE system. *dBC III's* C functions let you extend existing dBASE applications or allow your users to process their data using *dBC III* or dBASE III.

Lattice Text Utilities (TMU™) — *$75.00*

Lattice Text Utilities consists of eight software tools to help you manage your text files. GREP searches files for the specified pattern. DIFF compares two files and lists their differences. EXTRACT creates a list of file names to be extracted from the current directory. BUILD creates batch files from a previously generated file name list. WC displays the number of characters and optionally the checksum of a specified file. ED is a line editor which can utilize output from other *TMU* software in an automated batch mode. SPLAT searches files for a specified character string and replaces every occurrence with a specified string. And FILES lists, copies, erases or removes files or entire directory structures which meet the specified conditions.

Lattice Unicalc® Spreadsheet — *$79.95*

Unicalc is a simple-to-operate program that turns your AMIGA computer into an electronic spreadsheet. Using *Unicalc* you can easily create sales reports, expense accounts, balance sheets, or any other reports you had to do manually.

Unicalc offers the versatility you've come to expect from business software, plus the speed and processing power of the AMIGA.

• 8192 row by 256 column processing area • Comprehensive context-sensitive help screens • Cells can contain numeric, algebraic formulas and titles • Foreign language customization for all prompts and messages • Complete library of algebraic and conditional functions • Dual window capabilities • Floating point and scientific notation available • Complete load, save and print capabilities • Unique customization capability for your every application • Full compatibility with other leading spreadsheets • Full menu and mouse support.

Lattice MacLibrary™ — *$100.00*

The *Lattice MacLibrary™* is a collection of more than sixty C functions which allow you to quickly and efficiently take advantage of the powerful capabilities of the AMIGA.

Even if your knowledge of the AMIGA is limited, *MacLibrary* can ease your job of implementing screens, windows and gadgets by utilizing the functions, examples and sample programs included with the package.

Other *MacLibrary* routines are functionally compatible with the most widely used Apple® Macintosh™ Quickdraw Routines™, Standard File Package and Toolbox Utility Routines enabling you to rapidly convert your Macintosh programs to run on the AMIGA.

Panel™ — *$195.00*

Panel will help you write your screen programs and layer your screen designs with up to ten overlapping images. *Panel's* screen layouts can be assigned to individual windows and may be dynamically loaded from files or compiled into a program. *Panel* will output C source for including in your applications. A monitor and keyboard utility is also included to allow you to customize your applications for other systems.

With Lattice products you get *Lattice Service* including telephone support, notice of new products and enhancements and a 30-day money-back guarantee. Corporate license agreements available.

Lattice

Lattice, Incorporated
Post Office Box 3072
Glen Ellyn, Illinois 60138
(312) 858-7950 TWX 910-291-2190

INTERNATIONAL SALES OFFICES: Benelux: Ines Datacom (32) 2-720-51-61
Japan: Lifeboat, Inc. (03)293-4711 England: Roundhill (0672)54675
France: SFL (1)46-66-11-55 Germany: Pfotenhaur (49)7841/5058
Hong Kong: Prima 85258442525 A.I. Soft Korea, Inc. (02)7836372

Amazing Mail:

Modula-2 Problems and Concerns

Dear Amazing Computing™
A few comments on Mr. Faiwiszewski's review of TDI's Modula-2 in issue 9:

- No mention was made of Modula-2's alphabetic case sensitivity. The sensitivity will come as a rude shock to Pascal programmers. C programmers are accustomed to the absurdity of case-sensitivity, but they too will be confused at times since Modula's rules are almost exactly the inverse of C's.

- Although Mr. Faiwiszewski identified the problem, he was quite kind toward TDI regarding the documentation deficiency. One can get the feeling that one is playing Hacker when trying to determine what language features TDI has implemented and how they implemented them.

- Distributing updates via bulletin boards may be a positive attribute for those in major population centers, but it is not for those of us beyond local calls to the networks. TDI's mail support is non-existent and their telephone support is on such limited hours as to be impractical.

- Mr. Faiwiszewski failed to mention several fundamental deficiencies of Modula-2, including, but not limited to;
  Modula-2 has no intrinsic varying length character string data type. This deficiency can be somewhat circumvented, but there ain't no substitute for the real thing.
  Functions can return only simple values (no arrays or records). FORTRAN 66 programmers are well aquainted with the difficulties this limitation can cause when trying to produce character-string functions.
  Modula has no intrinsic I/O capabilities. The defects in standard Pascal's I/O capabilities are well known, but at least it has some capabilities to be defective. Modula evades the issue by providing nothing.

- The statement in the article that standard Pascal performs full expression evaluation is inaccurate. The ISO and ANSI/IEEE Pascal standards state that "short-circuit" or full expression evaluation is an implementation option. Most Pascal implementations perform full expression evaluation so Mr. Faiwiszewski statement is not wrong.

Sincerely yours,
Everett M. Greene
Ridgecrest, CA

Dear Amazing Computing™,
There are a couple of corrections and additions I would like to make to my Public Domain Modula-2 article in the last issue (Volume 2, #1, page 51):

In the text box on page 54, it is possible to use the "all" flag with delete to delete an entire subdirectory and its contents in one step. This makes clearing off the disk much easier and faster. For example, the fonts directory (which is not really necessary) can be deleted with the statement:

```
Delete fonts all
```

This is a much easier operation than deleting all the files and subdirectories in that directory first.

In that same text box, a carriage return got dropped. The last sequence of commands should look like this:

```
Copy * to s/Startup-Sequence
Assign M2: :M2
CD M2
Stack 10000
NewCLI "CON:0/100/640/100/One Pass Modula-2"
ALoad
```

I hope this clears things up.

Sincerely,
Warren Block

Dear AC:
I look forward to each issue of Amzing Computing with great enthusiasm, especially since you've covered Modula-2 so well.

As a Modula-2 programmer , I would like to see a Modula-2 implementation of the heap, radix, quick, and address calculation sorts with intelligent interface to select the quickest sort for whatever data is supplied by the user.

Sorting accounts for a lot of computer time, and I think that the programmers in your audience would find a program of this type extremely useful.

Thank you,
Vernon Dale Frameli
Houston, TX

Corrections:
Keep Track of Your Business Usage for Taxes (Vol. 1#9)

Dear AC:
The startup-sequence procedure in the article "Keep Track of Your Business Usage for Taxes" (AC, Vol.9) does not quite function properly when used with Version 1.2 of Kickatart and Workbench. For some reason unknown to the author, the date/time line and the name/usage-code line are concatenated into one line in the s/usage file under Version 1.2. In this situation, the AmigaBASIC program to print the usage statistics will not work right.

Fortunately, the fix is fairly simple. Actually, there are two options. For the first option, you may simply end each line with a carriage return and the 'CTRL-\'. This means that you have to add an extra keystroke, the carriage return, for the date/time line and for the name/usage-code line. If you choose this option, then you should edit the s/startup-sequence file prompt line to read:

```
ECHO "TERMINATE NEXT TWO ENTRY-LINES WITH 'CR' AND 'CTRL-\'"
```

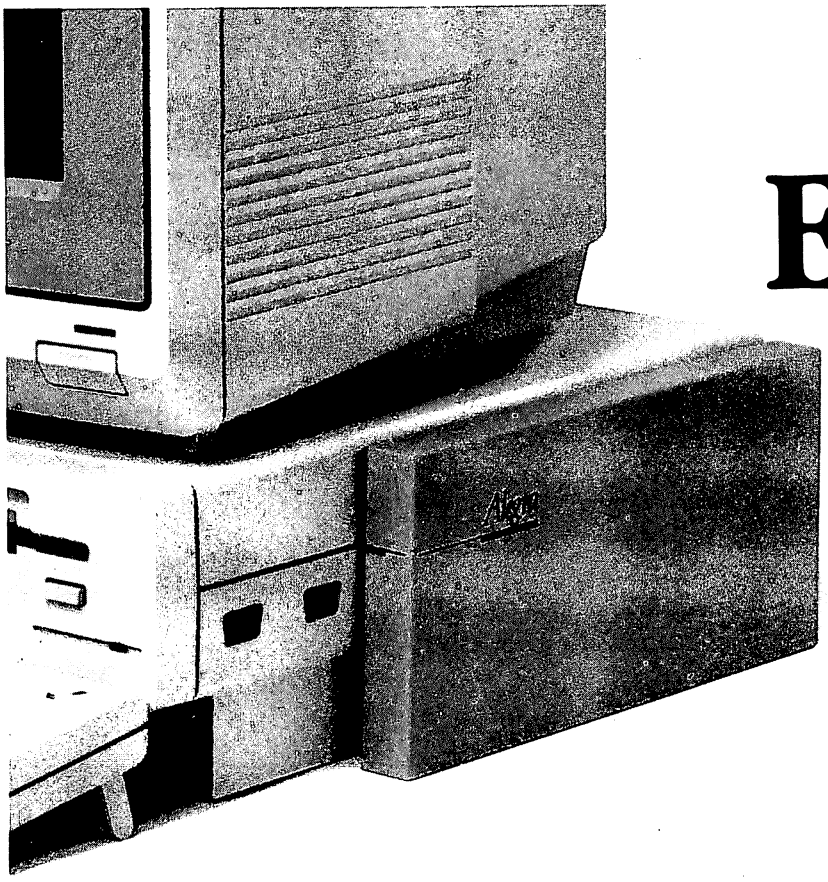
The other option does not require the extra keystrokes every time you boot up your computer. In this option, edit the JOIN line in the s/startup-sequence file to read:

```
JOIN RAM:time4 RAM:time1 RAM:time3
s/crf RAM:time2 AS SYS:s/usage
```

In order for this second option to work, you must have first created the file s/crf as an empty file. To do this, from your CLI, simply type ED S/CRF, and then close the file using <ESC>X<CR>. With this empty file interposed between the time3 and time2 RAM: files in the JOIN command, the date/time and the name/usage-code will appear on separate lines in the s/usage file. Whichever option you choose to employ, the new startup-sequence lines described in the article should be added just above the LoadWb line. Leave the two 'if EXISTS' line-groups and the 'BindDrivers' line intact.

Regards,
Jim Kummer
Lakewood, CO

•AC•



Expansion Memory Without The Wait.

Introducing *Alegra*: The Amiga™ Memory Expansion Unit from Access Associates.

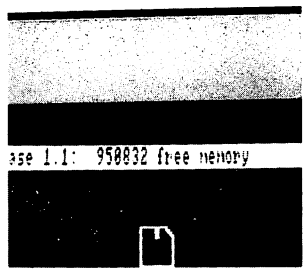
512 K now.

Now you can add 512 K bytes of external memory to your Amiga. In the smallest package available, a footprint only ¾"-wide. And Alegra's no-wait-state design lets your Amiga operate at its intended speed. No delays. With Alegra you get the benefit of fast memory at a surprisingly economical price. AND, BEST OF ALL, IT'S AVAILABLE NOW.

Upgradeable to 2 MB later.

If you'll need 2 MB of memory in the future, Alegra is still the right choice now. Our 2 megabyte upgrade (using 1 megabit DRAMs) will give you the memory you need in the same compact package.

Ask for Alegra at your quality Amiga dealer.



Total system memory is approximately 1 meg with the addition of our 512 K Alegra (depending on specific hardware configurations).

ACCESS ASSOCIATES

491 Aldo Avenue
Santa Clara, CA 95054-2303
408-727-8520

Thank You!

From the entire staff at

Amazing Computing

Floris Gamble

Mark Thibault

Kevin Dismarais

Robby

Bob Gamble

Keith McCormick

John David Eastin

Joyce A. Hicks

Ernest P. Yarmuth Jr.

John Foust

The Modem

One Peripheral Every Amiga Owner Should Have

"... if you really want to see what your Amiga can do, get yourself a modem and go on line. You'll be glad you did."

By Joseph L Rothman
President Amiga Mouse Users Group (A.M.U.G.)
BBS Telephone # 516-234-6046

In December 1985 I decided to take the plunge and buy an Amiga. Like many of my fellow Amiga owners, all I could do was look at the few demos I had received from my dealer. I was really impressed with what I saw, but I got bored very quickly. There was nothing for me to do but watch and listen.

AbasiC was no big thrill either. I was accustomed to a full screen editor and I absolutely refused to program in AbasiC.

When Dos 1.1 came out, things got better. I could now stare at the Electronic Arts Polyscope demo for two hours at a time. At least I could try my hand at programming using AmigaBasic, which is excellent, but since I am not much of a programmer, AmigaBASIC was not enough to satisfy my craving either. Programs started to appear on the market and I bought many of them, but they were few and far between and I remained bored. I needed something that would let me really use the Amiga on a regular basis, something that would not get boring.

Telecommunications, The Answer

I found out about BBS-PC! for the Amiga, so I called Micro Systems Software, got more information, and decided to start an Amiga Bulletin Board Service (BBS). I reasoned that if I started a BBS it would not only give me something useful and interesting to do with my Amiga, but it would also provide a service to the Amiga community and hopefully stimulate Amiga sales. I decided to make it 100% Amiga, for and about the Amiga only. A powerful computer like the Amiga deserves a 2400 baud modem, which I am using.

After I bought the modem, it took about three weeks to get the extra phone line installed and to properly set up in order to go on line. On March 10th, 1986, I opened my electronic doors and the Amiga Mouse Users Group (A.M.U.G.) was born. It was not long before I discovered that I wasn't alone, there were several other Amiga BBS's already. I was in for many more pleasant surprises from that day onward.

Public Domain Paradise

There is an amazing amount of Public Domain software for the Amiga and one way to access it, is by modem.

You may say, "Public Domain? If it was any good someone would have sold it." This is simply not true. The Amiga presents a challenge to programmers. The attraction to Amiga's features is incredible. Many programmers want to share their accomplishments, so they release their software creations to the Public Domain.

There are literally hundreds of useful utilities, games, new Dos commands, and productivity software programs just waiting for you on a local BBS. If art or music is your cup of tea, you can find more picture and song files than you will know what to do with. In addition to all of this software, you will also find new fonts, languages, printer drivers for printers that are not on the Dos disk and exciting demos of soon to be released products.

If that is not enough, the educational benefit of owning a modem is highly valuable. If you have a question about anything that has to do with using or programming the Amiga, you will most likely be able to find the answer through a message left on an Amiga BBS.

BBS VS. Telecommunications Service

When you have a modem you have many choices of where to call. As of this writing there are over 250 Amiga related BBS telephone numbers in the A.M.U.G. BBS list. BBS's are located all over the U.S. and Canada. There are also telecommunication services such as People Link and Compuserve which have extensive Amiga sections.

The major disadvantage of using these commercial services is the cost of an hourly charge for connect time. Another disadvantage is that these services either do not support 2400 baud (effectively doubling the cost of the phone call and line charges) or they do support 2400 baud and charge extra for it.

On a BBS the most you will ever have to pay is a small membership fee and the cost of the phone call. Not all BBS's charge for membership either, some of them are run by stores whose payoff is the exposure the store gets from the BBS and the mail order business the BBS generates.

Contributions Make The Difference

Whether or not the BBS charges for access, you should plan on contributing something. Your contribution can be in the form of financial aid for privately run boards, Public Domain software or just plain sharing of information. After all, running a BBS is a very expensive proposition. Not only is there the cost of equipment and phone line to consider, but the System Operator (SYSOP) spends a great deal of time maintaining the board and answering messages.

Don't get me wrong, I enjoy every minute of it, but my point is that the SYSOP's contributions are not enough. He cannot always afford things that make the BBS better for all members. Items such as hard drives to increase storage space for informative articles, the message base and the file section plus extra memory to run the board from ram (thereby increasing the speed of the BBS and decreasing the cost of the user's phone call) are expensive. It is the users of a BBS that determine how good the board is. Uploads of Public Domain software and information are just as important as operating capital. The more the users contribute, the better the BBS becomes.

Getting Started

The world of telecommunications may be new to you but don't let that stop you from going on line. Deciding what modem to buy is fairly easy, just make sure it is fully Hayes compatible. You can get a 1200 baud modem for less than $200 or a 2400 baud modem for around $400. The extra money spent now on 2400 baud will later be saved on your phone bills if you plan on doing a lot of interstate BBS hopping. If a service you intend to access doesn't support 2400 baud you can adjust to 1200 baud at any time under program control and return to 2400 whenever you want.

Installing a modem correctly is relatively simple. Most modems come out of the box ready to use, as does most terminal software. Hook up a modular phone jack to the modem and one cable from the modem to the back of the Amiga, then plug the modem power cord into an electrical outlet, turn everything on, load the software, set the baud rate and you are ready to go. If you have a problem you can get help from your dealer, a local users group, or a friend.

Going On Line

Going on line for the first time may seem scary but it is really easy. Read the terminal program's instructions on how to dial a phone number, then dial a phone number of a local BBS (check out the listing in this issue). Then follow the prompts to log into the system. With most BBS's the choices are on menus, just pick what you want to do.

The first thing you should do is enter yourself into the permanent user file, which will increase your access to the system immediately or with a delay of either a day or two during which the SYSOP confirms your application. You can join as many boards as you like.

If you really want to see what your Amiga can do, get yourself a modem and go on line. You'll be glad you did. A few minutes of connect time with a BBS can give hours of enjoyment by providing you with a useful new program or getting you the information you need to solve a problem. A modem might very well prove to be your most useful Amiga Peripheral.

•AC•

Amazing Reviews...

MacroModem

by Kent Engineering & Design

"...stands out from the rest, because of the complexity of the macros that one can create."

Reviewed by
Stephen R. Pietrowicz

USEnet: ...!ihnp4!pur-ee!gould!houligan!srp
PeopleLink: CBM*STEVE

A "macro" is a small string of characters that is expanded by the software interpreting it into a larger more complex command. Macros are used by assemblers and some compilers to let the user create smaller and more easy to type commands to replace longer, more complex commands.

In telecommunications programs, macros are used to send often-repeated commands to the remote computer system. The user simply defines the macros they want to use, logs in, and uses them while online.

MacroModem is another entry into the telecommunications software market for the Amiga, from *Kent Engineering & Design.* As the name implies, the MacroModem software allows the user to create macros for use while online to a work environment, BBS systems, or national computer networks. While many other programs have macro abilities, Macromodem stands out from the rest, because of the complexity of the macros that one can create.

The package consists of the MacroModem manual, a quick reference strip that rests in the small indented space above the function keys on the keyboard, a registration card, and two reader feedback cards. The disk includes the MacroModem program, the MacroEditor, and FileFilter. The software is not copy protected, and can run on either version 1.1 or 1.2 of AmigaDos. The manual consists of 5 chapters, and two appendices.

System Setup

Chapter one of the manual explains how to create your disk, depending on your particular system configuration: Workbench 1.1 or 1.2 and one or two disk drives. After creating the disk, it explains how to install the software. The user is prompted to answer questions about their preferences of where to store data that is captured from a remote system, how the modem should be initialized, window size, etc.

After answering which macro library and phone library to use, the user can save the default settings, and never have to worry about them again. The rest of the chapter explains how to set up the system for the first call to a remote host.

MacroModem is unconventional in the use of windows, menus, and the mouse. MacroModem has no menus that other Amiga programs do; it uses mouse controlled windows to make the Macro and Help windows appear. The left mouse button turns the Macro window on and off, and the right mouse button turns the Help window on and off.

Function Keys

Chapter two explains how to use the pre-programmed function keys. The function keys are programmed with twenty commands:

Key	Command	Key	Command
F1	Receive file	Shift-F1	Toggle CRC
F2	Send file	Shift-F2	Send Text file
F3	Open Capture	Shift-F3	Close Capture
F4	Capture On/Off	Shift-F4	Read Capture
F5	Compose Start/Send	Shift-F5	Perform NewCLI
F6	Set Path	Shift-F6	Directory On/Off
F7	Set Baud	Shift-F7	Serial Setup
F8	Macro Help	Shift-F8	Load New Macro File
F9	Auto Dial	Shift-F9	Load New Phone File
F10	Command mode	Shift-F10	Quit

Most of the above commands are self-explanatory, but some might need further explanation:

continued...

F1/F2
Files uploaded and downloaded using these commands use the XMODEM protocol.

F3/Shift-F3/F4
Text that is captured during a session with a remote host are controlled by these keys. A file must first be specified before data capture can take place (by using F3). To close the file use Shift-F3. F4 is used to toggle whether capture mode is on or off. A user can only use this to capture certain messages or files for use later. It allows the the user to read the capture buffer within the program.

F5
Lets the user compose up to 10 lines of text to transmit while in a conference.

F6
Sets the directory in which data is stored.

F7
Shows a directory listing of the current directory listing program for this function. The user can substitute their favorite directory listing program for this function.

Up to 36 additional macros can be programmed by the user (any key A-Z or 0-9). Macros can be "chained" together, so one macro can branch to another macro. This can be used quite effectively to create very complex macros. One of the macro libraries included with the package has such a macro. It will dial the phone and log into one of the national computer services with just one keystroke.

Macros can also load other macro libraries to replace the current library. For example, an initial macro set can perform all of the functions of logging into one of several communication services. After selecting the service to log into, the macro can bring in the macro library that is used on that service.

One of the best things about MacroModem is the ability to select any of the macros using either the mouse or the keyboard. The macros can be activated by typing <ALT>-<macro_key> or by selecting it with the mouse from a window. If you sent up the macros correctly and use the mouse, you perform most online functions without even touching the keyboard.

A complaint I have about this chapter in the manual is the tendency to repeat things. Each reference to entering a filename in any of the commands that use them, the same text is repeated, over and over. Although it serves as a nice reminder when making quick references to an individual command, it was quite irritating when I was reading about each command for the first time.

Command mode

Chapter three explains commands that can be issued after the user presses the F9 key and places MacroModem into "*Command Mode*". Command mode allows the user to change different terminal modes easily without having to use the Terminal mode of the program. (MacroModem is in

Terminal mode when it executes macros). Command mode options are split into 6 categories:

"*File commands*" are used to maintain local system files and directories. Commands to load in new macro sets, create directories, etc. are in this category.

"*Setup commands*" are used to change default terminal mode settings and macros. Adjustments to parity, baud rate, control code masking (to mask out unprintable characters), etc. are made with these commands.

Once while I was trying to set different baud rates, I tried to set the baud rate to its maximum rate (262,000). I entered the command "262000 baud" at the command prompt, and then doing a STAT to see if the baud rate was set correctly. It wasn't. Instead, entering the command as "262,000 baud" worked. (Note the comma). But if the user isn't aware of this, it can be confusing.

"File transfer" options are in category 3. Open/close capture, file transfer commands, etc. are explained here.

"Command mode exits" lists commands that let the user switch from command mode to terminal mode (or quit the program). Upon entering terminal mode, the user can keep default echo/linefeed settings or change them, depending on their needs.

"Call Automation" lists commands for a simple script language. As the manual states, MacroModem isn't meant to be used as a completely automated terminal program. The script language only has commands to wait for strings and respond to them. It is possible for the user to set up commands to automatically log into a system. Once logged in they can issue commands to the system (or run their own macros) to do any of the remote system functions.

"Miscellaneous commands" include ways to call Preferences, start a NewCli, hooks for programs like DirUtil, and also the SHELL" command.

MacroEditor

Chapter 4 gives an introduction to macros, explains the features and operation of the MacroEditor, and leads the user through a sample edit session using the MacroEditor.

When MacroEditor starts, it displays two windows. One window is used to display the macros themselves, and the other window is used to display the help messages for the macros.

MacroEditor is quite easy to use. To edit a macro, the user clicks on the letter of the macro they want to edit. A small edit window will appear. One line shows the "help" message, in which the user types a descriptive message about the macro, and the next line shows the macro itself. The Macro can contain just about any keystroke. The unusual keystrokes (such as function keys) are displayed differently so the user can distinguish them from regular keystrokes.

Creating different new macros and changing existing macros is somewhat difficult at first, simply because MacroEditor allows so many different keystrokes within a macro. The edit window provides gadgets to delete or insert characters. When a mistake is made in editing a macro, the natural tendency is to type backspace; if backspace is typed, however, it will be entered as part of the macro. The delete gadget has to be turned on in order to delete characters. After entering macros in the edit window for a while, it is easier to remember, but it takes a while to get used to it.

FileFilter

MacroModem's designer decided not to include some features in MacroModem because they add to "command clutter". By "command clutter" he means that he wanted to avoid having "so many options to choose from that it becomes counterproductive". He also cites the problem of downloading a program, not being sure of what form it is in, and then having to re-download the file to try a different method of chopping or filtering. In order to help the user, he created "FileFilter" which is described in Chapter 5.

FileFilter has a variety of options. It allows the user to change the end of line characters of text files, to remove form feeds from text, and to filter only ASCII characters from files. It can also expand tabs to spaces, or insert form feeds (or do both at once). FileFilter also chops files automatically and can chop files to certain lengths.

The chopping options are welcome for a new user who has very little experience with downloading files. However, more experienced users tend to have less problem with downloading files. MacroModem has the autochop feature built-in, and will prompt the user whether or not to chop the file. It's really a nice option to have, especially for a new user. Considering the amount of problems that almost all users have downloading strange text files at one time or another, those options are quite useful.

Summary

The manual for MacroModem is somewhat disorganized, and at times a bit confusing. However, the tutorials are well written, and the software itself outshines the problems I encountered with the manual. Macros are very easy to write and to test, and most importantly, the program accomplishes its goal: To let the user obtain the information they want from the service quickly and easily without wasting time, and to do that without a lot of effort.

•AC•

MacroModem $69.95

Kent Engineering & Design
Box 178
Mottville, NY 13119
(315) 685-8237

GEMINI

or "It takes two to Tango"

"... the concept of competing against someone located at the other end of a telephone line, whether it's next door or across the country (if you have the money for the phone bill)."

by Jim Meadows

Two-computer Games

For quite some time I have wanted to be able to call up a friend and compete in some type of computer game, each using our own computer. After awhile I decided to start the ball rolling and try to write such a game. The constellation GEMINI means "The Twins". So I chose GEMINI as an appropriate name for my two-computer game series. So far I have written two games (two programs make a series don't they?). I wrote GEMINI-1 in 1985 on the IBM PC mainly to demonstrate the concept and generate some interest in two-computer games. It was basically a real-time strategy game that used only character graphics for the display. After receiving a sufficient response to GEMINI-1, I decided to write a more sophisticated 3D tank game that you could play with a friend over the telephone ... assuming he also had a computer and a modem.

I first developed GEMINI-2 for the IBM PC since I already had my communication techniques worked out for it. However, it was also about this time that I purchased my AMIGA, and knew that as soon as I could, I would convert GEMINI-2 to the AMIGA and make use of the AMIGA's advanced sound and graphics capabilities. You can in fact run GEMINI-2 on the AMIGA and play someone running GEMINI-2 on their IBM PC, but what a difference AMIGA's hardware makes for special effects!

Knowing that there may be others interested in the concept of two-computer games, I decided to write this article to pass on what I have run across so far in my programming and communications efforts. I have also included a basic sample program (I'll call it GEMINI-.5) that you can use to try out the concept of competing against someone located at the other end of a telephone line, whether it's next door or across the country (if you have the money for the phone bill).

Information Exchange

What issues must be considered when developing two-computer games? The first issue to resolve is deciding what information must be exchanged between computers. You are going to have the same program running on two computers at the same time and you must determine what information is needed to be transmitted between computers in order for both to duplicate what happens on the other computer. One approach would be to pass each user's input on to the other computer (e.g. key presses, mouse movements, etc.). For the sample program and the GEMINI programs I have chosen to pass results rather than inputs to insure each computer exactly tracks what happens on the other computer. This however brings up other questions such as, which results should be computed and which results are to be received from the other computer? When two programs are running in parallel, you don't want them to both be making the same decisions (i.e. did your shell hit his tank). Only one computer should make the decision and notify the other of the results. Which one decides what? Do you check to see if your tank shell hit his tank and transmit the results, or do you check to see if your tank was hit by his shell and transmit those results?

In the sample program I chose for the program to always compute if it hit the opponent's tank and to be notified by the other computer if your tank is hit. This insures that when you see you have hit the other person's tank, he also records that he has been hit.

Information Format

What form should the information be in to transmit between computers? For the sample program I chose a fixed format containing 7 bytes in each transmission as illustrated below:

1	2	3	4	5	6	7
Your Tank Row	Your Tank Col.	Your Tank Dir.	Your Shell Row	Your Shell Col.	Hit Flag (Score)	End of Trans. (CR)

Your current tank row and column location and direction on the game grid is sent in every transmission as well as your shell row and column location. If you have not fired recently and you have no shell in flight, values off the game grid will be transmitted to indicate to the receiving computer you

have no shell to display. The hit flag is always zero except when you have computed that you hit your opponent's tank. The hit flag is then set to your new score. Whenever you receive a non-zero hit flag from your opponent, you know you have been hit and you also have his new score. I chose in this example for the end of transmission character to simply be a carriage return so that each transmission string "looks" like a 7 byte text line. In order for the row, column and other values to never be confused with a carriage return (or any of the other control characters 0-31) an offset of 32 is added to each value by the transmitting computer and subtracted from each value by the receiving computer. This allows each byte to have a value range from 0 to 95 when using 7 data bits with parity in your transmission (i.e. 127-32=95).

Instead of a fixed length transmission, you could have a beginning control byte to indicate what information was included in the transmission, and have the transmission only include values that have changed. However by keeping the transmission length fixed you maintain a constant transmission time as well as having simpler programming.

Transmit/Receive Synchronization

Another issue to resolve is, when do you transmit and receive information? The first program I tried I thought, "I'll only transmit when I have results to transmit (i.e. when something changes)." I quickly found out this can lead to synchronization problems where the two computers might not stay together in what they think is going on. You also run into problems this way if the computers are running at different speeds -- the faster one gets ahead of the other one.

One solution that I have found to be quite effective is to always be exchanging information whether anything has changed or not. This effectively keeps both computers synchronized regardless of difference in processing speeds (e.g. An AMIGA can play an IBM). A generalized processing sequence is shown below that illustrates this:

*1. **Compute***
- *a. Get your input (if any)*
- *b. Compute results of input*

*2. **Transmit***
- *a. Build transmission string from program variables*
- *b. Send transmission string to other computer*

*3. **Receive***
- *a. Get bytes transmitted from other computer*
- *b. Check for valid transmission*
- *c. Convert received bytes to program variables*

*4. **Display***
- *a. Perform any special processing of received data*
- *b. Display results of your input and received data*
- *c. Go back to step 1*

By always transmitting each time through the program loop and waiting to receive a transmission from the other computer before proceeding, you force the two computers to effectively stay synchronized so that one does not get ahead of the other one.

Communication Errors

What happens though when you get an error while receiving a transmission and how do you check for errors? There are various communication errors you can check for in your program. If you set up your communications to use even or odd parity, the hardware will perform some error checking for you and you can use the ON ERROR command in basic to trap parity errors. However this will limit you to 7 data bits in each byte of your transmission. You can ignore parity and get the full 8 bits per byte if you wish. I choose to adapt my communications to 7 data bits per byte and make use of hardware parity checking. Another possibility is something happening that keeps you from getting a complete transmission. You don't want your computer to wait indefinitely since this could result in a deadlock situation (both computers waiting on one another). Therefore it is a good idea to put a time-out and exit from your receive wait state and continue, assuming something has gone wrong.

You can also get quite sophisticated in performing some of your own error checking after you have received a transmission. You could use a special identifier at the start

and/or end of the transmission that must always be present to have a valid transmission. If each transmission should include a certain number of bytes this could be verified after receiving the end of transmission character. You could also include a check byte at the end of the transmission that is a function of all the other bytes in the transmission. The receiving computer could then perform this function on all the bytes received and see if it matches the check byte. The function could be as simple as exclusive-or'ing together all bytes received or as involved as using CRC error checking and correction algorithms. You may wish to make certain that there are no more characters in the receive buffer after you have received the end of transmission character in case something has thrown off the synchronization between transmissions.

Besides determining if you have received a valid transmission, you need to decide if the transmitting computer needs to know if the other computer received a good transmission or not. The receiving computer could send an acknowledge code (ACK) or a negative acknowledge code (NAK) to the transmitting computer to let it know whether the information was received correctly or not. In sophisticated communication schemes (such as IBM's SNA communications), the receiving computer may tell the transmitting computer which portion of the communications was received in error so that it can be sent again by the transmitting computer. This is critical for applications such as data communications networks, but may not be necessary for the game environment.

One approach that I tried when I first started experimenting with developing two-computer games was to have the receiving computer transmit back each character received so that the transmitting computer knew the character was received OK. This is a very slow method of communications, so I quickly changed to grouping the information and sending it together with a termination character to denote the end of the transmission. I also tried sending back an ACK/NAK response and retransmitting the information if a NAK was sent. However I began to realize that if you are transmitting results rather than user input, it didn't matter too much if you had to ignore a bad transmission -- you would still get the opponent's correct positions on the following transmission. So I dropped the ACK/NAK response totally which greatly sped up the communications process. You need to evaluate this for each application though since it may not always be feasible to ignore a bad transmission.

Establishing Communications

When I began trying out my first two-computer program I had an APPLE sitting next to an IBM and had their serial ports connected together using a null modem adapter. Since I could see both screens at the same time it wasn't too difficult to know if I had gotten the two computers "talking" to each other correctly or not. However when I separated them I realized that when they are connected using modems over phone lines I needed a way to make sure a valid communication link had been established (both using the same baud rate, parity, etc.). Plus I needed a simple way to dial or answer using the modem. What easier way is there to verify your communication link than to "talk" to your opponent using the keyboard before starting the game? If

you are receiving garbage on the screen perhaps you are using different baud rates. If some characters are OK and others are not, you may not have the same parity settings. If each character you typw shows up twice on the screen, your modem is in echo mode and should be changed (e.g. enter ATE0 to change a Hayes modem). If all characters look OK, you have a good communications link!

Therefore I added a simple terminal routine at start-up to allow you to issue commands to your modem as well as converse with your opponent until you are both ready to begin. The terminal mode would end when one of the players pressed the "start-the-game" key (I chose to use the $ character for this). When someone pressed this key, it was also transmitted to the other computer to signal it to end its terminal mode and also start the game. This also solved the problem of uniquely identifying Player A from Player B. Whoever pressed the $ key could be placed at the top of the game grid and whoever received the $ characer could be placed at the bottom, etc.

```
10 '    *****  GEMINI-.5  *****
20 ' An Example of a Two-computer Game
30 '    by Jim Meadows   12/86
40 '
41 '  This program may also be run on
42 ' an IBM PC by putting an apostrophe
43 ' at the beginning of the lines noted
44 ' by <-- Amiga and removing the
45 ' apostrophe from the lines noted
```

```
46 ' by <-- IBM. (See lines 50-85,
47 ' 1012-1050, and 20020-20030)
48 '
50 SCREEN 1,320,200,2,1      :' <-- Amiga
60 WINDOW 2,"GEMINI-.5",,,1  :' <-- Amiga
70 PALETTE 2,1,0,0           :' <-- Amiga
72 PALETTE 3,0,1,0           :' <-- Amiga
75 COLR1=1:COLR3=3:BSK=8     :' <-- Amiga
80 ' KEY OFF:SCREEN 1        :' <-- IBM
82 ' COLOR 1,3               :' <-- IBM
85 ' COLR1=3:COLR3=1:BSK=29  :' <-- IBM
97 '
98 ' --- Establish Communications ---
99 '
100 CLS:' (Display Instructions)
110 PRINT "          *** Gemini-.5 *** "
115 PRINT
120 PRINT " 1. Establish Communications "
130 PRINT "     Type  ATDT123-4567  to dial"
140 PRINT "     or    ATA          to answer"
145 PRINT
150 PRINT " 2. Check Communications"
160 PRINT "     Enter  HELLO , etc."
165 PRINT
170 PRINT " 3. Start Game "
180 PRINT "     ONE of you press $"
190 PRINT "     (press Esc to Quit)"
191 PRINT
194 '
195 GOSUB 10000:' (Initialize serial port)
198 '
199 ' (Receive and display opponents chars)
200 IF EOF(1) THEN 300
205 T$=INPUT$(1,#1)
210 IF T$<>CHR$(29) AND T$<>CHR$(8) THEN 220
211  T$=CHR$(BSK):PRINT T$+" ";:' (Backspace)
220 PRINT T$;:' (Print Character)
230 IF T$=CHR$(13) THEN SOUND 900,1:' (CR)
240 IF T$=CHR$(27) THEN 20000:' (Esc)
250 IF T$="$" THEN PLAYER$="B":GOTO 400
298 '
299 ' (Read keyboard and display)
300 K$=INKEY$:IF K$="" THEN 200
310 IF K$<>CHR$(8) AND K$<>CHR$(29) THEN 320
311 K$=CHR$(BSK):PRINT K$+" ";
320 PRINT K$;:PRINT #1,K$;:' (Print/Transmit)
330 IF K$=CHR$(13) THEN SOUND 1000,1:' (CR)
340 IF K$=CHR$(27) THEN 20000:' (Esc)
350 IF K$="$" THEN PLAYER$="A":GOTO 400
360 GOTO 200
397 '
398 ' ---- Game Initialization ----
399 '
400 CLS
401 LOCATE 1,13:PRINT "** Gemini-.5 ** "
405 WIN=15:' (Winning Score)
410 NUMCOL=5:COLW=40:' (Column size)
420 NUMROW=4:ROWH=30:' (Row size)
425 COLO=40:ROWO=10:' (Upper left offsets)
428 OS=32:' (Comm. byte offset value)
430 ' Variable prefix U=You, O=Opponent
440 USCORE=0:OSCORE=0:' (Reset Scores)
450 USHCOL=NUMCOL+1:USHROW=NUMROW+1:' (shell)
460 OSHCOL=NUMCOL+1:OSHROW=NUMROW+1
465 IF PLAYER$="B" THEN 480
469 ' Player A
470 UTKCOL=0:UTKROW=0:' (tank)
472 OTKCOL=NUMCOL:OTKROW=NUMROW
474 UTKDIR=3:OTKDIR=1:' (Tank directions)
478 GOTO 500
479 ' Player B
480 OTKCOL=0:OTKCOL=0:' (tank)
482 UTKCOL=NUMCOL:UTKROW=NUMROW
484 OTKDIR=3:UTKDIR=1:' (Tank directions)
498 '
499 ' Draw Game Grid
500 ENDROW=NUMROW+1:ENDCOL=NUMCOL+1
505 FOR I = 0 TO ENDCOL
510 X=COLO+I*COLW:Y=ROWO
520 LINE (X,Y)-(X,Y+ENDROW*ROWH)
530 NEXT
540 FOR I = 0 TO ENDROW
550 Y=ROWO+I*ROWH:X=COLO
560 LINE(X,Y)-(X+ENDCOL*COLW,Y)
570 NEXT
580 LOCATE 22,5:PRINT "Your Score: "
590 LOCATE 22,25:PRINT "Opp. Score: "
596 ON ERROR GOTO 0
597 '
598 ' ---- Main Program Loop ----
599 '
600 GOSUB 1000:' Compute
610 GOSUB 2000:' Transmit
620 GOSUB 3000:' Receive
630 GOSUB 4000:' Display
640 GOTO 600
997 '
998 ' ---- 1. Compute ----
999 '
1000 ' (Check for keyboard input)
1001 K$=INKEY$:IF K$="" THEN 1200
1009 ' (Only use last key pressed)
1010 X$=INKEY$:IF X$<>"" THEN K$=X$:GOTO 1010
1012 IF K$=CHR$(28) THEN 1110:' (Up)    <-- Amiga
1014 IF K$=CHR$(30) THEN 1120:' (RIGHT) <-- Amiga
1016 IF K$=CHR$(29) THEN 1130:' (DOWN)  <-- Amiga
1018 IF K$=CHR$(31) THEN 1140:' (LEFT)  <-- Amiga
1020 ' IF K$=CHR$(0)+CHR$(72) THEN 1110:'(Up) <-- IBM
1030 ' IF K$=CHR$(0)+CHR$(77) THEN 1120:'(Right)<-IBM
1040 ' IF K$=CHR$(0)+CHR$(80) THEN 1130:'(Down)<--IBM
1050 ' IF K$=CHR$(0)+CHR$(75) THEN 1140:'(Left)<--IBM
1060 IF K$=" " THEN 1150:' (Fire)
1070 IF K$=CHR$(27) THEN 1160:' (Esc)
1080 K=0:SOUND 100,1:' (Invalid key)
1090 GOTO 1200
1100 '
1109 '
1110 IF UTKDIR<>1 THEN UTKDIR=1:GOTO 1200
1115 UTKROW=UTKROW-1:IF UTKROW<0 THEN UTKROW=0
1118 GOTO 1200
1119 '
1120 IF UTKDIR<>2 THEN UTKDIR=2:GOTO 1200
1125 UTKCOL=UTKCOL+1
1126 IF UTKCOL>NUMCOL THEN UTKCOL=NUMCOL
1128 GOTO 1200
1129 '
1130 IF UTKDIR<>3 THEN UTKDIR=3:GOTO 1200
1135 UTKROW=UTKROW+1
1196 IF UTKROW>NUMROW THEN UTKROW=NUMROW
1138 GOTO 1200
1139 '
1140 IF UTKDIR<>4 THEN UTKDIR=4:GOTO 1200
1145 UTKCOL=UTKCOL-1:IF UTKCOL<0 THEN UTKCOL=0
1148 GOTO 1200
1149 '
1150 USHROW=UTKROW:USHCOL=UTKCOL
1152 USHDIR=UTKDIR
1155 SOUND 2000,1
1158 GOTO 1200
1159 '
1160 PRINT #1,K$:' (Send Esc to opponent)
1165 GOTO 100
1169 '
1200 IF USHCOL>NUMCOL THEN 1300
1210 ON USHDIR GOTO 1220,1230,1240,1250
1219 '
1220 USHROW=USHROW-1:IF USHROW<0 THEN GOSUB 1290
1221 GOTO 1300
1230 USHCOL=USHCOL+1:IF USHCOL>NUMCOL THEN GOSUB 1290
1231 GOTO 1300
1240 USHROW=USHROW+1:IF USHROW>NUMROW THEN GOSUB 1290
1241 GOTO 1300
1250 USHCOL=USHCOL-1:IF USHCOL<0 THEN GOSUB 1290
1251 GOTO 1300
```

```
1290 USHCOL=NUMCOL+1:USHROW=NUMROW+1
1292 RETURN
1300 RETURN
1997 '
1998 ' ---- 2. Transmit ----
1999 '
2000 T$=CHR$(OS+UTKROW)+CHR$(OS+UTKCOL)
2010 T$=T$+CHR$(OS+UTKDIR)
2020 T$=T$+CHR$(OS+USHROW)+CHR$(OS+USHCOL)
2030 T$=T$+CHR$(OS+TSCORE)
2040 PRINT #1,T$+CHR$(13);
2050 TSCORE=0
2200 RETURN
2997 '
2998 ' ---- 3. Receive ----
2999 '
3000 T$=""
3005 ON ERROR GOTO 10100
3010 TMOUT=VAL(RIGHT$(TIME$,2))+5
3015 IF TMOUT>59 THEN TMOUT=TMOUT-60
3020 IF NOT EOF(1) THEN 3100
3030 IF VAL(RIGHT$(TIME$,2))<>TMOUT THEN 3020
3032  LOCATE 1,1:PRINT "No Response"
3033  SOUND 800,3
3034  LOCATE 1,1:PRINT "            "
3035  GOTO 3500
3100 T$=T$+INPUT$(1,#1)
3110 IF RIGHT$(T$,1)=CHR$(27) THEN 100
3120 IF RIGHT$(T$,1)<>CHR$(13) THEN 3020
3130 IF EOF(1) THEN 3200
3132  LOCATE 1,1:PRINT "Over-run"
3133  SOUND 900,3
3134  LOCATE 1,1:PRINT "            "
3135  T$=""
3138  GOTO 3020
3200 IF LEN(T$)=7 THEN 3300
3232  LOCATE 1,1:PRINT "Bad Length"
3233  SOUND 1000,3
3234  LOCATE 1,1:PRINT "            "
3235  GOTO 3500
3300 OTKROW=ASC(MID$(T$,1))-OS
3310 OTKCOL=ASC(MID$(T$,2))-OS
3320 OTKDIR=ASC(MID$(T$,3))-OS
3330 OSHROW=ASC(MID$(T$,4))-OS
3340 OSHCOL=ASC(MID$(T$,5))-OS
3350 OSCORE=ASC(MID$(T$,6))-OS
3500 ON ERROR GOTO 0
3510 RETURN
3998 ' ---- 4. Display ----
3999 '
4000 ' (Check if your shell hit his tank)
4005 IF USHCOL<>OTKCOL THEN 4100
4010 IF USHROW<>OTKROW THEN 4100
4020 USCORE=USCORE+1
4030 TSCORE=USCORE
4100 ' (Check if you can see his tank or shell)
4105 OTKSEE=0:OSHSEE=0
4110 ON UTKDIR GOTO 4120,4130,4140,4150
4120 IF UTKCOL=OTKCOL AND OTKROW<UTKROW THEN OTKSEE=1
4121 IF UTKCOL=OSHCOL AND OSHROW<UTKROW THEN OSHSEE=1
4122 GOTO 4200
4130 IF UTKROW=OTKROW AND OTKCOL>UTKCOL THEN OTKSEE=1
4131 IF UTKROW=OSHROW AND OSHCOL>UTKCOL THEN OSHSEE=1
4132 GOTO 4200
4140 IF UTKCOL=OTKCOL AND OTKROW>UTKROW THEN OTKSEE=1
4141 IF UTKCOL=OSHCOL AND OSHROW>UTKROW THEN OSHSEE=1
4142 GOTO 4200
4150 IF UTKROW=OTKROW AND OTKCOL<UTKCOL THEN OTKSEE=1
4151 IF UTKROW=OSHROW AND OSHCOL<UTKCOL THEN OSHSEE=1
4152 GOTO 4200
4200 ' (Blank out tank and shell old locations)
4205 COL=UTKCOLO:ROW=UTKROWO:GOSUB 4290
4215 COL=USHCOLO:ROW=USHROWO:GOSUB 4290
4225 COL=OTKCOLO:ROW=OTKROWO:GOSUB 4290
4235 COL=OSHCOLO:ROW=OSHROWO:GOSUB 4290
4240 UTKCOLO=UTKCOL:UTKROWO=UTKROW
4250 USHCOLO=USHCOL:USHROWO=USHROW
4260 OTKCOLO=OTKCOL:OTKROWO=OTKROW
4270 OSHCOLO=OSHCOL:OSHROWO=OSHROW
4280 GOTO 4300
4290 IF COL>NUMCOL OR ROW>NUMROW THEN RETURN
4292 X=COLO+COL*COLW
4294 Y=ROWO+ROW*ROWH
4296 LINE(X+1,Y+1)-(X+COLW-1,Y+ROWH-1),0,BF
4298 RETURN
4300 ' (Display Tanks and Shells)
4305 TCOL=COLR3
4310 COL=UTKCOL:ROW=UTKROW:DIR=UTKDIR
4315 GOSUB 4380
4320 COL=USHCOL:ROW=USHROW
4325 GOSUB 4390
4330 TCOL=2
4340 COL=OTKCOL:ROW=OTKROW:DIR=OTKDIR
4345 IF OTKSEE=1 THEN GOSUB 4380
4350 COL=OSHCOL:ROW=OSHROW
4355 IF OSHSEE=1 THEN GOSUB 4390
4360 GOTO 4400
4380 IF COL>NUMCOL OR ROW>NUMROW THEN RETURN
4381 X=COLO+COL*COLW+COLW/2
4382 Y=ROWO+ROW*ROWH+ROWH/2
4384 LINE (X-COLW/4,Y-ROWH/4)-
(X+COLW/4,Y+ROWH/4),TCOL,BF
4385 ON DIR GOTO 4386,4387,4388,4389
4386 LINE(X,Y)-(X,Y-ROWH/2+2),COLR1:RETURN
4387 LINE(X,Y)-(X+COLW/2-2,Y),COLR1:RETURN
4388 LINE(X,Y)-(X,Y+ROWH/2-2),COLR1:RETURN
4389 LINE(X,Y)-(X-COLW/2+2,Y),COLR1:RETURN
4390 IF COL>NUMCOL OR ROW>NUMROW THEN RETURN
4391 X=COLO+COL*COLW+COLW/2
4392 Y=ROWO+ROW*ROWH+ROWH/2
4394 CIRCLE (X,Y),COLW/8,TCOL
4396 RETURN
4400 ' (Update score for any hits)
4405 IF TSCORE=0 THEN 4450
4410 LOCATE 22,16:PRINT TSCORE
4420 SOUND 1000,1:SOUND 1500,5
4430 GOSUB 1290
4440 IF TSCORE<WIN THEN 4450
4445 LOCATE 12,15:PRINT "You Win!!"
4446SOUND1000,1:SOUND1500,2:SOUND1000,1:SOUND1500,10
4447 GOSUB 2000:FOR I = 1 TO 3000:NEXT
4448 GOTO 100
4450 IF OSCORE=0 THEN 4500
4460 LOCATE 22,36:PRINT OSCORE
4470 SOUND 1500,1:SOUND 1000,5
4480 IF OSCORE<WIN THEN 4500
4485 LOCATE 12,15:PRINT "You lose!"
4487 SOUND 1500,3:SOUND 1000,5:SOUND 500,10
4488 FOR I = 1 TO 4000:NEXT
4489 GOTO 100
4500 ' (End of Display Routine)
4510 RETURN
9998 ' Initialize Serial Port
9999 '
10000 CLOSE
10005 OPEN "COM1:" AS #1:' (Open serial port)
10010 ON ERROR GOTO 10050
10020 RETURN
10049 '
10050 PRINT "Comm. Error":SOUND 2000,1
10060 RESUME 195
10099 '
10100 LOCATE 1,1:PRINT "*** Comm. Error ***"
10110 SOUND 2000,1
10120 LOCATE 1,1:PRINT "                    "
10150 RESUME
19998 ' Quit
20000 PRINT "Disconnect":BEEP
20005 ON ERROR GOTO 0
20010 FOR I = 1 TO 4000:NEXT
20020 WINDOW CLOSE 2        :' <-- Amiga
20030 SCREEN CLOSE 1        :' <-- Amiga
20040 END
```

•AC•

We bring the AMIGA to life...

A STEREO sound digitizer that every Amiga owner should have!

PERFECT SOUND

by SunRize Industries

FUN: RECORD SOUNDS, PLAY THEM BACK FASTER OR SLOWER, EVEN PLAY THEM BACKWARDS

AFFORDABLE: ONLY **$79.95**

APPLICATIONS: ADD SPEECH, MUSIC AND SOUND EFFECTS TO YOUR PROGRAMS

SPECIFICATIONS:

- Two channel digitizer that *records* in *stereo*
- Includes superb editing software
  - IFF compatible
  - Instrument editing
  - "C" source code
  - Graphs, file compression, and more
- Typeset manual
- Library of recorded sounds
- Free technical support

MICROSEARCH

9896 Southwest Freeway
Houston, Texas 77042

(713) 988-2818

Dealer Inquires Invited

Amazing Reviews...

BBS-PC!

by Micro-Systems Software

"...integrates all of the elements of a good BBS system into one package."

Reviewed by
Stephen R. Pietrowicz

Eight years ago, when I first started dialing up to BBS systems, I was fascinated. Just by picking up a phone, dialing the BBS number, and putting the handset into the acoustical coupler, I was able to use someone's computer. I decided I would one day run a BBS system for myself.

Since then, I'd used many different computer systems and had not been on the local BBS circuit. When I finally bought my Amiga, I realized the best source of information about the machine were the Amiga BBS systems that were being run by users just like myself. BBSing helped me learn more about the Amiga.

I also realized I finally had the resources to set up my own BBS system. After calling the phone company to install the second phone line, I waited. The day I received BBS-PC! in the mail, I was finally able to run my own BBS system.

BBS-PC! integrates all of the elements of a good BBS system into one package: messages bases, file transfer capabilities, private mail, security features, support for open or closed BBS systems, and much more. It is made by Micro-Systems Software, the makers of the Online! telecommunications program.

Setting up the system
The minimal BBS system requires 512K and one drive. Two drives are recommended. Included with the software are a manual, and a "quick setup" guide. The "quick setup" guide explains how to the BBS that they have set up on the disk running. The guide suggests the new system operator (or SYSOP, for short) use this for a while to get used to it, and to examine the examples carefully if the SYSOP wants to create their own menus.

If you've never run a board before, it's a very good idea to follow this advice. Test the sample configuration for a while. Once you feel comfortable with it, look at the menu definitions. Reading the menu definition really helps the novice SYSOP understand the "inner workings" of the BBS. Follow the instructions very carefully. Missing one data file when you move them to where you want them in the file system will cause the BBS to fail to boot.

The only problem I had with the set up provided on the disk was in the "Apply for membership" command. When a new user tries to use this command the system prompts for the SYSOP password. A quick look in the reference guide corrected the problem. I simply gave the command the right access code, recompiled the menu and it all worked fine.

The BBSINIT utility included in the package is used to fully configure the system. BBSINIT prompts for the location of the directories that contain membership information, caller information, file upload/download information and the message base. Once this is run, and all the system files are in place, the system is ready for the first caller.

Once the system is booted, the "wait for caller" menu shows the number of callers, the total number of messages (along with the low and high message number) the number of files in the file area, and the files to be approved by the SYSOP.

BBS-PC! functions
BBS-PC! has 54 different functions to help maintain the system, as well as perform commands for users. These functions are used in the menu definition to specify what action is to be taken when a certain key is pressed. The functions are listed in the manual in the table of contents, with a description and an example in section 3 of the manual. Functions are organized in several different categories: General functions, Message functions, file transfer functions, SYSOP functions, Communication functions and Control functions.

General functions are functions used to check/change to user statistics, the system section names, time, etc.

Message functions are used to read, write, scan, and delete messages.

Steve Pietrowicz works as a member of the technical staff of Gould Electronics, Computer Systems Division in Fort Lauderdale, Florida. He is also an assistant SYSOP on PeopleLink's Amiga Zone. He also runs "The Amiga Resource" BBS (300/1200) (305)-486-8849.

File transfer function are used to let the user upload, download, read, and kill files.

Sysop functions are used to maintain the user data base and set up user, terminal, modem and node defaults.

Communication functions are used to maintain change the system dialout capabilities (if a second modem is available), and direct file transfers.

Control functions are used to control how menus and bulletins are presented to the user.

Some parts of the manual reference function 111, a command to allow the user to execute a system command. The function does not appear in the table of contents, and doesn't appear with the rest of the functions in the book. A call to the MSS board confirmed that function 111 doesn't exist in the Amiga version of BBS-PC!.

Menus

The part of BBS-PC! I like the most is the user definable menu structure. The menus are completely configurable. The BBS-PC! includes a "menu compiler" that takes a simple language as input, and builds your menus.

The SYSOPs are able to design the system menus any way they want. Some SYSOPs prefer to have numbered commands, while others use the first letter of the command to execute. Even the style of the menus themselves can be changed. If all the users on a BBS have 80 column capability, the SYSOP takes full advantage of that. If a SYSOP wants to design special 40 column menus for the users that only have 40 columns, that can be done too. Both sets of menus can be maintained by the same BBS, and switched depending on how the SYSOP sets access codes.

When the SYSOP defines the menus they must give each command a privilege range. This is used to restrict user access to certain BBS functions. The SYSOP can set up the BBS to restrict the access of guests to the system until they have a chance to look over their user statistics.

When a user logs in and receives the menu on the screen, the BBS checks for the user's access code as it prints out the menu. If the user's privilege is within the range specified in the menu definition, the command will be printed to the screen and the user can execute that command. If the user doesn't have access to a command, but knows it exists and tries to execute the command, the BBS will tell him the command is invalid.

The SYSOP also has the ability to make certain menus appear for certain users by carefully choosing the privilege codes for each user. For example, a BBS might be setup for Commodore 64, 128 and Amiga Users. The SYSOP decides to set the privilege codes for the C64 owners to 64, codes for the C128 owners to 128, and the codes for Amiga owners to 150. When setting up the menus, the SYSOP specifies that the "M" command will get the user into the message base, but only for those messages that the user is interested in. This can all be done by simply specifying that certain access codes will perform certain operations.

The SYSOP can setup the BBS so certain users have SYSOP capabilities in sections. This is really a nice feature, especially if the traffic in your BBS is large and you need the extra help.

Message base

The message base allows for 16 different sections and they are completely definable by the SYSOP. The appearance of messages is also definable by the SYSOP. When the SYSOP sets up a new section, he must decide which flags to set for that section. Flags incorporate things like the date, message sender, addressee, ability to delete the message after it has been read, etc.

In sections that the SYSOP wants to use for electronic mail, there is a flag for personal messages. No one other than the sender and recipient can read the messages when the flag is set. One problem I found with the message base is that users can send mail to non-existent user IDs. Someone once called my BBS and sent mail to "Steve", and it didn't complain at all. If I hadn't been watching the BBS at that time, I would never have known there was a message in the system for me. I called the MSS BBS about this, and was told it may be corrected in a future release of the software.

There are some really nice features incorporated into the message system. If you address the message to one specific person, the message is marked with an "(X)" indicating the message has been read by that person.

If a certain topic carries on for several messages, BBS-PC! will let you follow the message "thread". That's one of the most useful features on a BBS that is heavily used. It's much easier to keep your train of thought on one specific topic, rather than reading messages in chronological order.

Even though there are 16 sections, I didn't use them all when I set up my BBS to leave room for expansion in the future.

File section

The file section is organized in the same way as the message section. Each message section can have its own download section. The SYSOP can configure the system to display files by section, alphabetical order, aged order, or by natural order. An aged order display allows users to scan the most recent files added to the system. There are two different ways files can be presented to the user. Catalogs display the file names, file sizes, the file dates and descriptions. Browsing files lets the user display one file at a time, and include the user's name that uploaded the file.

BBS-PC! offers four file transfer methods: XMODEM, XMODEM-CRC, Hayes Smartcom, and ASCII transfer. When a user uploads a file, the SYSOP has the option to have it placed into a special directory until it is approved. The system shows that files are waiting approval at the "wait for caller" menu. When the SYSOP logs into the board, the system will also announce that files are waiting approval.

One of the things I didn't like about the file section was the room allowed for description is limited to 40 characters. It's hard to describe some files well enough so the users of the system know what the file does. After all, it's the only way users have of knowing if a file is worth the time to download.

System Utilities
There are 6 utilities to help the SYSOP run the BBS:

BBSINIT creates the BBS.P file that contains the system configuration. BBSINIT prompts the user for the locations of files on the system. It's possible to use the extra disk drives or hard disk SYSOP might have by telling BBSINIT where you want everything stored.

BBSMENU is the BBS menu compiler. It takes definitions of menus as input from a text file, and creates the files necessary for the BBS menus.

BBSINFO is a use report generator. The SYSOP enters the average time per day that the BBS runs, the name of the previous report file, and the name of the report you want to generate. Once this is done, BBSINFO generates the file STATS.TXT. STATS.TXT contains an average hourly usage chart, average percentage of section usage chart, communications statistics such as how many users used 300 baud, 1200 baud, and 2400 baud, and more. It's really quite interesting reading, and can help the SYSOP maintain the system according to user needs.

BBSFIX is used to help rebuild the system if it gets corrupted, as by a power failure.

BBSFILE is used by the SYSOP to do mass uploads to the file sections of the BBS. Otherwise the SYSOP would have to do this one file at a time, and that can take quite a while.

CHKFILE is used to cross reference files that are actually on the system with files in the database. Files that aren't popular can be kept offline to help save space, but will still appear in the file listing.

System Security
BBS-PC! offers 256 levels of access codes. As I explained above, the access codes can be used to restrict or permit access to certain sections of the board. This enables the SYSOP to make the board as complicated as he wants to make it. In using the system myself, I've found that 3 access levels are all I really use: Guest (someone who dials up for the first time), Member, and SYSOP. According to Micro-Systems, to this date, no BBS-PC! system has ever been crashed. That's quite a feat!

Other Sections
The manual includes a brief section titled "Principles of Operating a BBS". It contains advice for the novice SYSOP, but is good reading for even a more experienced SYSOP.

Appendix A of the manual explains the file structure scheme used by BBS-PC! for those SYSOPs out there who want to write their own BBS utilities. Appendix B contains information on Hayes compatible and other modems. It goes into detail about how the SYSOP can set up a modem correctly.

All in all, I've enjoyed setting up the board, and watching people use the system. BBS's do require a lot of time to maintain, but the time put into it is well worth it. If you do set up a BBS, be sure and spread the word that your system is up and running. Post messages about it everywhere you can. Ask the users that call into the system what THEY want to see on your BBS. At first this may be difficult, but in time you'll get your own loyal user base.

BBS-PC! $99.95

Micro-Systems Software, Inc.
4301-18 Oak Circle
Boca Raton, FL 33431

BBS-PC! requires 512K and one disk drive. Two drives are recommended.

MSS also maintains a BBS. It runs 24 hours a day, 7 days a week, at 300, 1200 and 2400 baud, at (305) 737-1590.

•AC•

DATABASE
SPREADSHEET
WORD PROCESSOR

Micro-Systems Software —Bigger and Better

for the Amiga

NEW - ORGANIZE! DATABASE MANAGER, VERSION 1.0

Mailing lists! Club memberships! Patient records! Client files! Video tape libraries! Phone call logs! Nearly anything that needs to be filed, sorted or calculated is a candidate for **Organize!**

In seconds, **Organize!** can scan your files, locate information, and display or print in the format you want. Use it to print form letters with the Mailmerge function of Scribble!. Or calculate fields and do statistical analyses of your files with many of the same built-in math functions from Analyze!.

Easily design input forms and output reports with the mouse and pull-down menus. Just as simply - store, sort, review and print. The file size is limited only by disk space and the format is compatible with the industry standard dBASE format.

End your paper shuffle! Get **Organize!** today.

Only $99.95

UPGRADED - ANALYZE! SPREADSHEET, VERSION 2.0

ANALYZE! FEATURES:

- Pulldown menu interface (mouse-driven).
- Large spreadsheets with efficient memory usage.
- Dedicated function keys for common commands.

NEW ADDITIONS:

- **Business Graphics**; print bar, stacked bar, pie graphs in 2 or 3-D; line, X-Y, area graphs; all in 4 or 8 colors; data from spreadsheets; IFF format; view up to 4 graphs at same time; instantly redraw graphs when data changes; ranges, labels, titles, legends, rotation, scaling; fast and effective!
- **Command Macros**; save keystrokes; create templates.
- **Sorting**; rearrange row or column data quickly.
- File Icons; access spreadsheets via icons or names.

NEW PRICE Only $149.95

IMPROVED - SCRIBBLE! WORD PROCESSOR, VERSION 1.0

SCRIBBLE! FEATURES:

- Pulldown menu interface (mouse-driven)
- Multiple windows; edit/cut & paste 4 documents on screen.
- Preview; see final form on screen before printing.

NEW ADDITIONS:

- **Spellcheck**; expandable 40,000 word dictionary; check word, all words on screen, or entire document; alternative spellings shown.
- **Mailmerge**; print form letters, mailing labels; create data file with Scribble! or Organize!
- File Icons; access documents via icons or names; copy documents by pulling icons across workbench.
- Expanded Memory Support; for larger documents.
- More Amiga Keys; menu commands from keyboard or mouse.
- More Flexibility; Wordstar™ commands; scrolling while cut/paste; improved file operations.

Still only $99.95

MICRO-SYSTEMS SOFTWARE, INC.

4301-18 OAK CIRCLE, BOCA RATON, FL 33431
IN FL. CALL (305)391-5077 VISA, MASTERCARD

For Nearest Dealer Call 1-800-327-8724

See your local dealer or call:

Brown-Wagh Publishing

1-800-451-0900

1-408-395-3838 *(in California)*

16795 Lark Ave., Suite 210, Los Gatos, CA 95030

Wordstar is a trademark of Micropro International. Amiga is a registered trademark of Commodore-Amiga, Inc.

Amiga Bulletin Board Systems

by Joe Rothman

This is a list of other BBS's throughout the U.S. and Canada. It was compiled as a cooperative effort by the following BBS Sysops:

Amiga America	619-364-3816	Sysop	Chet Solace
A.M.U.G.	516-234-6046	Sysop	Joe Rothman
PSA-BBS	414-278-5390	Sysop	Dorothy Dean

All listings that have a C next to the phone number have been confirmed by the A.M.U.G. BBS SYSOP or another reputable source. If a B appears in that column it means there was a busy signal every time I tried to call that number. An N means I recieved no answer on all my attempts to confirm that number. Please keep in mind, these numbers may become inactive at any time. Please use discretion when dialing unconfirmed numbers for the first time. If you find an unconfirmed number is correct, or you are able to provide more information, please leave a message to one of the above SYSOPs . Conversely if you find out that any information contained herein is incorrect please let them know.

Joe (SYSOP) Rothman

A.M.U.G. BBS 516-234-6046

LOCATION	NAME	NUMBER	CONF'D	HOURS	BAUD	NOTES
USA						
AK	North Lights	907-337-4136	C		3/12	
AZ	Bob's Mach	602-242-3158	C		3/12	
AZ	MasterCom	602-244-8096	C		3/12	
AZ Phoenix	Amiga Talk	602-846-3901	C	24 HOUR	3/12/24	New #
AZ	Lost Horiz.	602-849-0110	C		3/12	
					C128	W/AmigaSec.
AZ	EZ Rider	602-935-4680	C	9PM-7AM	3/12	
AZ	Space Sta.	602-978-4430	C		3/12	
CA Fresno	RBBS	209-229-9589	C		3/12/24	
CA	AGE BBS	209-438-0510	C		3/12	BBS-PC!
CA L'Angels	Thru Glass	213-325-0213	C			
CA L'Angels	AmigaBoard	213-478-9788	C		3/12/24	BBS in Beta
CA	Flying Bug	213-839-6867	C		3/12	
CA	Starfleet	408-244-1614	C		3/12	S,Amiga,Sec.
CA	X-BBS	408-336-2249	C		3/12	
CA	Pyrzqxgl	408-336-3134	C		3/12	
CA	Stuart II	408-338-9511	C		3/12	Amiga Sec.
CA	Ace BBS	408-353-4531	C		3/12	Amiga Sec.
CA	Devel Xchg	408-372-1722	C		3/12	MaxiCorp Bd.
CA	THE WELL	415-332-6106	C		3/12	$2/H + $8/Mo
CA	Amiga West	415-355-7162	C		3/12	
CA Castro V	Sphinx Soc	415-581-9452	C		3/12	Amiga Sec.
CA S'Carlos	F A U G	415-595-2479	C	6PM-7AM	3/12	24 H WEEKENDS
CA	RSVP-BBS	415-659-9169	C		3/12	Fidonet node
CA	Big City Ni	415-863-1781	C	6PM-6AM	3/12	
CA	BBS-JC	415-961-7250	C	24 HOUR	3/12/24	Amiga sec.
CA	Highlands	619-298-6307	C		3/12	
CA Landers	Amiga Amer.	619-364-3816	C	24 HOUR	3/12	Start 2/11/7
CA	Alexandrian	619-576-7424	C		3/12	
CA Del Mar	Dreamscape	619-755-2863	N	24 HOUR	3/12/24	
CA	AMIC	707-579-0523	C	24 HOUR	3/12	Huge Lib.
CA	Coast BBS	707-964-7114	C	24 HOUR	3/12	
						Erase 1st O at Name
CA S'Clmnte	VIVID XPRES	714-493-6094	N	24 HOUR	3/12	
						BBS source available.
CA	C.U.T.E.	714-532-5698	C	24 HOUR	3/12	
CA	Kingdom 3	714-653-2302	C		3/12	
CA Csta Msa	Amiga's Den	714-646-2723	B	24 HOUR		
CA Irvine	Action	714-660-1462	C	24 HOUR	3/12	AMIGA Sec.
CA	Amiga Line	714-772-4097	C		3/12	Color Many files
CA	Soft Celler	714-772-9671	C		3/12	Amiga Sec.
CA	Fontana	714-823-3673	C		3/12	Amiga Sec.
CA	Kingdom 1	714-829-8908	C		3/12	
CA	Myco-D	714-832-8016	C		3/12	Amiga Sec.
CA	Learning Pl	714-849-8332	C		3/12	Amiga Sec.
CA	Kingdom 2	714-886-0324	C		3/12	
CA	Amiga Xpres	805-397-4812	N	24 HOUR	3/12	
CA Ventura	VTBBS	805-656-3746	C	24 HOUR	3/12/24	Amiga sec.
CA	Abacus	805-832-7186	C		3/12	$ Membership
CA S'Barbra	Compucation	805-967-0895	C	24 HOUR	3/12/24	Amiga sec.
CA Bkrsfld	Amy Project	805-834-9383	C	24 HOUR	3/12	BBS-PC! Amiga
CA	L. Frontier	818-345-2918	C		3/12	
CA	United Com.	818-982-8495	C		3/12/24	
CA	Elec. Mag.	916-662-9591	C		3/12	
CA	Hot Tub	916-689-4670	B	24 HOUR	3/12	Except Sun.Nt
CA Sacra		916-933-4329	N	24 HOUR	3/12	
						Written in MS Basic
CO	Buck Board	303-427-9539	C		3/12	TBBS Amiga Sec.
CO Boulder	Fido	303-497-6968	C		3/12	Amiga Sec.
CO Pueblo	Towne Crier	303-545-8480	C	24 HOUR	3/12	Amiga sect. D
CO Pueblo	M.A.E. BBS	303-564-0618	C	9PM-4AM	3/12	Mountain Time
CO CO Sprgs	SHAMUS ST	303-591-8673	C	24 HOUR	3/12	Amiga/ST bd
CO Denver	TBBS	303-693-4735	C	24 HOUR	3/12/24	
CO	Grotto	303-694-9050	C		3/12	TBBS,Amiga Sec.
CO	Mile High	303-730-2250	C		3/12	
CO	Stampede	303-799-9733	C		3/12	ST, Amiga Sec.
CO	Neutral Zone	303-985-9184	C		3/12	
CO	UnknownTBBS	303-988-8155	C	24 HOUR	3/12	Amiga Sec.
CT	Brand-X-Fido	203-255-7729	C		3/12	Amiga Sec.
CT	Shoreline	203-468-2853	C		3/12	Fido,Amiga Sec.
CT	Skyline	203-488-0816	C		3/12	Atari,Amiga Sec.
DE	Scholars	302-451-8045	C		3/12	
DE	Beagle Nest	302-731-7842	C	10P-10A	3/12/24	sec. 6
DE Wilmngtn	PC-NUG Brd	302-737-2294	C	24 HOUR	3/12	Amiga sec.21
FL B Raton	Nightline	305-368-5837	C	6PM-8AM	3/12/24	24 H WE
FL	ARIA	305-435-9837	C		3/12	
FL	Fortune C.	305-439-2136	C	12P-8AM	3/12	Amiga sup.
FL Ft. Laud	AMIGAMAIL	305-486-7021	N	9PM-8AM	3/12	MSS BBS-PC
FL	Resource	305-486-8849	C		3/12	
FL Cocoa	Online	305-636-2664	C		3/12/24	Spacecoast
FL	Bug Shop	305-641-1162	C		3/12/24	
						Computer Flea Market
FL	NiteOwl	305-588-0487	C		3/12	RBBS, Color
FL B Raton	MSS-HQ	305-737-1590	C	24 HOUR	3/12/24	
FL	DOSBBS	305-737-1644	C		3/12	

LOCATION	NAME	NUMBER	CONF	HOURS	BAUD	NOTES
FL	Doctor Fido	305-744-7862	C		3/12	Amiga Sec.
FL	Amiga Cave	305-798-3576	C	8PM-3AM		
FL	Amigalink	305-798-5928	C	3PM-Mid	3/12	24 Hrs WE
FL	C= Forum	305-832-7369	C		3/12	
FL	Bloom County	305-869-0226	B	24 HOUR	12	
FL	Amiga-BBS	813-924-2626	C		3/12	Computer Store
FL Tampa	"Mon Ami"	813-985-7624	C	9PM-7AM	3/12	
FL	Hobbit Hole	904-243-6219	C	24 Hour	3/12	BBS-PC!
FL	Data*West	904-262-9629	C		3/12	C64,AmigaSec.
FL Jacksnvl	CASA MI AMY	904-733-4515	C	24 HOUR	3/12	Fidonet node
FL	Orions Neb.	904-777-1295	C		3/12	C64,AmigaSec.
FL	Socrates	904-771-7140	C		3/12	Amiga Sec.
GA Atlanta	Amiga Net	404-843-1938	C	24 HOUR	3/12	Amiga Sec.
GA Atlanta	AmigAtlanta	404-968-5011	N	1PM-9PM	3/12	
GA Savanna	Tardis	912-247-2194	C	6PM-6AM	3/12	24 Hours WE
GA Savanna	VSC-RBBS	912-333-5975	C		3/12	Color,AmigaSec.
HI	Cyber Sys.	808-521-3306	C	24 HOUR	3/12/24	Amiga bd.
IA	The City	515-283-1902	C	24 HOUR	3/12	
IA	Amiga Zone	712-366-9747	C	24 HOUR	3/12	Tag BBS (PD)
IL Ch'paign	Hack's Anon	217-333-8301	C	24 HOUR	3/12	Amiga Section
IL	Amiga Doc	312-351-8815	C		3/12	
IL Chicago		312-383-9482	N			
IL	BBS 1984	312-841-2401	C		3/12	Color,Amiga Sup.
IL Chicago	BBS Chicago	312-842-1745	C		3/12	Animated Color
IL GlenElyn	Lattice	312-858-8087	C	24 HOUR	3/12/24	Fidonet
IL	Wovsed 200	618-378-2133	C	5PM-8AM	3/12	
						Use Username:GUEST
KS	Amig-Oz	316-283-9210	C	24 HOUR	3/12/24	
						BBS-PC! Ram: 3 Drive
KS	J&L Elec.	316-624-8068	C	6PM-9AM	3/12	
KS Wichita	Datalink	316-651-0365	C		3/12	BBS-PC! Amiga
MA Wakefield	Tos Tech	617-246-5876	C	7PM-8AM	3/12/24	
						8N1,Amiga Sec.
MA Lowell	Toolbox	617-692-5476	C	24 HOUR	3/12	Amiga Sec.
MA	Worchester	617-835-2626	C		3/12	
						7PM Fri to Midn Sun
MA Worchstr	Mind Link	617-853-7420	C	24 HOUR	3/12	Fidonet
MA Cambridg	The Window	617-868-1430	C	24 HOUR	3/12	BBS-PC!
MA	Wonderland	617-665-3796	C		3/12	Amiga Sec. 10
MA	IDCMP	617-769-8444	C		3/12	
MA Dracut	Cantelope	617-957-3921	C	24 HOUR	3/12	Amiga Sec.
MD	T.A.C.	301-445-3777	C		3/12	BBS-PC!
MD	Ami Exchange	301-439-6220	C	24 HOUR	3/12	Magazine
MD Baltimor	U Maryland	301-455-3630	C	24 HOUR	3/12	Amiga sec.
MD	Johnny's Pl	301-585-0725	C		3/12/24	
MD	Sunrise 80	301-666-5703	C		3/12	7E1 only
MD	AMIBBS	301-666-9109	C		3/12/24	
MD	Outpost	301-681-8371	C		3/12	
MD	Amiga Land	301-985-0174	C		3/12	
MI Novi	Novi Downld	313-348-4479	C	24 HOUR	3/12	
						348-4477 Voice 1st
MI Canton	Can Comp	313-459-6930	N	7P-11P		
MI	Slipped Disk	313-585-8315	C		3/12	
MI	M Net	313-994-6333	C	24 HOUR	3/12	Join Amiga
MI	Enterprise	517-372-6037	C		3/12	
MI G Rapids	Big Blue	616-245-6635	C	24 HOUR	3/12/24	Sup. Amiga
MI	Multi Board	616-335-5119	C		3/12	
MN Minn.	Base 2 sys	612-535-0568	C	24 HOUR	3/12/24	Amiga Sec.
MN Minn.	Mindtools	612-542-8980	C	24 HOUR	3/12/24	Sup.Amiga
MN	Comp. Place	612-869-3246	C		3/12	
MO St Louis	MDC RCC	314-232-6881	C	24 HOUR	3/12/24	
						McDonnall Douglas
MO Gir-deau	Palace BBS	314-335-4902	C		3/12	Online Games 8N1
MO St Louis	Voice of Am	314-961-8084	C	24 HOUR	3/12	BBS-PC!
MO St Louis	Cobra BBS	314-429-3189	C		3/12	Amiga Sec.
MO St Louis	Spiders Wb	314-576-6232	C		3/12	St, Amiga Sec.
MO St Louis	Starship	314-867-6950	B			
MO St Louis	Systems+	314-961-8084	C	24 HOUR	3/12	
MO	Mebbsie	816-474-1052	C		3/12	Amiga is sec. 14
NC Cary	Deep Thought	919-471-6436	C	24 HOUR	3/12/24	Soft Dist.
NC	Third Wave	919-556-7975	C		3/12	
NC	Missing Link	919-724-7526	C		3/12	
NC	MMS	919-779-6674	C	24 HOUR	3/12/24	Amiga sec.
NC	CompTel Ctr	919-832-7282	C	24 HOUR	3/12	
NC	Alein	919-945-2818	C	8PM-8AM	3/12	
ND	Frozen Banana	701-746-5932	C	24 HOUR	3/12/24	
NE Omaha	Wind Dragon	402-291-8053	C	24 HOUR	3/12	
						Fidonet 14/614
NH Keene	The Tardis	603-357-4306	C		3/12	
NH Laconia	Sanctuary	603-524-0136	C	24 HOUR	3/12	
NH Dover	The Tardis	603-749-1017	C		3/12	
NH	Amiga Exc.	603-749-3038	C		3/12/24	
NJ	Excalibur's	201-256-0691	C	24 HOUR	3/12	BBS-PC! on
NJ	Drew U II	201-377-8193	C		3/12	Amiga Sec.
NJ	Drew U I	201-377-8245	C		3/12	Amiga Sec.
NJ	MANX	201-542-2793	C		3/12	Compiler Sup.
NJ Iselin	JABS BBS	201-750-4409	C	24 HOUR	12	JAUG Board
NJ Norm Bch	1st Legal	201-793-0996	C	24 HOUR	3/12	On St,Amiga $12
NJ	AMLINE	201-864-0121	C	8PM-8AM	3/12	Run by Store
NJ	AGJ	201-886-8041	C		3/12	Amiga Sec.
NJ	Boardroom	201-994-5195	C		3/12	Amiga Sec.
NM Albuquer	RCPM	505-299-5974	C	24 HOUR	3/12/24	Amiga Sec.
NM	New Horizons	505-437-9117	C		3/12	
NM	Messila Val	505-524-8372	C		3/12	Big RCPM
NV	Hubert	702-322-8877	C		3/12	Many Files
NV	Com Addicts	702-731-3178	C		3/12	C64 Amiga Files

LOCATION	NAME	NUMBER	CONF	HOURS	BAUD	NOTES
NY New York	AMUSE	212-269-4879	C	24 HR	3/12/24	Fidonet 107/34
NY C.I.	A.M.U.G.	516-234-6046	C	24 HOUR	3/12/24	100% AMIGA
NY Floral P	LIBBS	516-326-8753	C	24 HOUR	3/12/24	
						50 MEG, w/Amiga Sec.
NY Brtwater	AMY ADVNTGE	516-661-4881	C		3/12	
						Amiga Advantage Bd.
NY Long Is.	Zeitgeist	516-689-3105	C	24 HR	3/12/24	
						Users,Developers
NY	5th Gneratn	716-377-3985	C	24 HOUR	3/12	
NY Buffalo	Soft. Sales	716-873-5321	C	9PM-9AM	3/12/24	24 Hrs WE
NY Bklyn	Gateway NRA	718-338-3501	C	5PM-8AM	3/12	24 Hrs WE
NY Flushing	CIA BBS	718-357-6760	C	24 HOUR	3/12/24	
						30 MegST Amiga Sec.
NY	App on-line	718-746-1140	C	24 HOUR	3/12/24	AmigaFiles
NY	Hangout	718-897-5578	C	24 HOUR	3/12/24	
						30 MegST, Amiga Sec.
NY	Micro Lair	718-967-4286	C	10A-10P	3/12	
NY	MUSICNET	914-442-4006	C		3/12	MIDI Bd,$75/Yr
NY	Conexus	914-561-0864	C		3/12	Amiga Sec.
NY	Doc's R us	914-668-3664	C	24 HOUR	3/12	$ Membership
NY	Eyes of K H	914-779-5886	B		3/12	
OH	CA-AUG BBS	216-341-4452	C		3/12/24	
OH	Firehouse	216-352-9499	C		3/12	C64,AmigaSec.
OH Akron	Akron Amiga	216-633-9658	C		3/12	
OH	LCE	216-644-0028	C		3/12	
OH Willobee	No. Coast	216-953-8057	C	24 HOUR	3/12	
OH Cincin.	Shareware	513-531-4654	C		3/12	Amiga Section
OH Dayton	AmigaBoard	513-898-0702	C	24 HOUR	3/12	$ Membership
OH	Icon Net	614-237-0285	C	24 HOUR	3/12	IBM All Amiga
OH Columbus	Earthrise	614-868-1100	C		3/12O	
OH	ICON-2	614-895-7209	C		3/12O	
OK Stillwa.	Frontier	405-624-2429	C	24 HOUR	3/12/24	Amiga Sec.
PA	DELCHUG	215-363-6625	C		3/12	TBBS,Amiga Sec.
PA	Phil Amiga	215-533-3191	C	24 HOUR	3/12/24	
PA Lebanon	Yatch Club	717-273-6704	C	24 HOUR	12/24	Ap.IIe AmigaSec
PA Lancastr	The Tusk	717-560-1750	C	24 HOUR	3/12	BBS-PC!on Amiga
RI	Amiga Forum	401-732-3286	C		3/12	Color Menus
SC	Crit. Mass	803-366-6235	C	24 HOUR	3/12/24	
SC	MegaBD. 3000	803-583-2364	N		3/12/24	
SC	Causers	803-796-3127	C	24 HOUR	3/12	
SC Simpson.		803-967-2202	N	7PM-Mid	3/12	
TN	Connection	615-868-7860	C	24 HOUR	3/12	
TN Nash	Comm BBS	615-885-4162	C	24 HOUR	3/12	Amiga Sec.
TN	Duck Pond	901-755-5330	C		3/12	
TX D/FW	Rising Star	214-231-1372	C		3/12/24	Fidonet Amiga Sec3
TX D/FW	DMUG	214-276-8902	C	24 HOUR	3/12/24	USMidiUsers
TX D/FW	S.C.O.P.E	214-288-1537	C	24 HR	3/12/24	8N1only 60Megs
TX D/FW	Excaliber	214-341-2775	C	24 HR	3/12/24	BBS-PC!ClrMenu
TX San Ant	Amy San Ant	512-684-3433	C		3/12	
TX	AMICOM	512-737-2552	C		3/12	
TX Houston	Phoenix	713-776-9704	C	9AM-7PM	3/12	
TX D/FW	Nomads Nook	817-926-8922	C		3/12	
TX El Paso	BBS El Paso	915-772-0827	C	3PM-9AM	3/12	24HR Wkd AmySec
TX Austin	AAA II BBS	512-339-7134	C	7PM on	3/12	Busi. #, Day
TX San Ant	A.S.A	512-684-3433	C	24 HOUR	3/12/24	
UT SaltLake	TechniSoft	801-264-8290	C	24 HOUR	3/12	Fidonet 15/464
UT	CPU]l[	801-277-3200	C		3/12/24	
UT	Armadillo	801-484-2766	C		3/12	
VA Fairfax	Empire I	703-352-1936	C		3/12	
VA	FCC-ARENA	703-644-8445	C		3/12	Use 7E1
VA	Empire II	703-978-0148	C		3/12	
VA Norfork	Colony Soft	804-625-1945	N			
VA Lynchbrg	Fido	804-846-1880	C		3/12	Amiga Sec.
WA	A-Link BBS	206-774-4735	C	24 HOUR	3/12	40MB on Amiga
WA Everett	PD Silo	206-775-2650	C	24 HOUR	3/12	ALT 206-347-0243
WA Seattle	Stonehenge	206-324-0830	C	5P-8A	3/12	24 Hrs Wkds
WA	Rand Access	206-582-0906	C	8PM-6AM	3/12	
WA	Bangor BBS	206-697-1465	C	7PM-11P	3/12	
WA		206-874-6381	C	24 HOUR	3/12	Fidonet node
WI Milwalk	PSA-BBS	414-278-5390	C	24 HOUR	3/12	
WI	Green Bay	414-435-7074	C		3/12	60Mg Amiga Sec
WI	SUE	414-762-6475	C		3/12	Mlt User $ Bd
WI Milwalk	Comm Link	414-784-2096	C	24 HR	12/24	30Mg, Amiga Sec
WI	FDL-BBS	414-921-8448	C		3/12	ST, Amiga Sec
WI	Amiga Zone	414-968-2462	C		3/12	
WI	OF The Fly	608-836-1180	C		3/12	Flea Mkt
WI	Soul Forge	608-837-0453	C		3/12	ST, Amiga Sec
WV	BBS-WV	304-636-9097	C		3/12	Amiga Sec
CANADA						
Manitoba	Micro Mart	204-589-4742	C		3/12	Amiga
Section						
Saskat.	Regina BBS	306-347-4493	C		3/12	Fido, Amiga
Section						
Alberta	Phase 4	403-258-0844	C	630P-8A	3/12	Mt Time, Ram:
Ontario	Info-Centre	416-296-1202	N	6PM-9AM	3/12/24	
Ontario	Gateway	416-433-7077	C	6P-10A	3/12	24 Hrs Wkd
Ontario	BloomBeacon	416-445-2169	C	24HR	3/12/24	AmigaBBS-PC!Ram
Ontario	TROFF	416-529-6710	C		3/12	
Hamilton	Track 36	416-544-0011	C	24 HR	3/12	30MG, Amiga Sec
Ontario	WDIX	416-668-2078	C	24 HR	3/12	Whitby Info Exc
N Brunswick	Amiga World	506-357-9660	C	24 HOUR	3/12	Not the Mag.
Quebec	AMNET	514-645-3730	C		3/12	
Quebec	Montreal	514-989-1567	C	24 HOUR	3/12/24	
Ottawa	Online	613-526-4141	C		12	
Ontario	OMX	613-731-3419	C		3/12	
Ontario	Connect2	705-652-3506	C		3/12	

The Trouble With Xmodem

"...binary files downloaded using Xmodem or Xmodem-CRC would not load into my Amiga."

By Joseph L Rothman
President Amiga Mouse Users Group (A.M.U.G.)
BBS Telephone # 516-234-6046

When I was new to Xmodem file transfers, I was quite surprised to find out that binary files downloaded using Xmodem or Xmodem-CRC would not load into my Amiga. All I got was "File is not an object module". Great, what does that mean?

Why the Problem Exists

An Amiga binary file is any file that can be run from CLI or Workbench without first running another language. The Xmodem file transfer protocol was originally written for CP/M machines that stored their files in 128 byte blocks. During an Xmodem file transfer, the two computers exchange these data blocks one block at a time, each machine checking the other to insure that the data is being received intact. If a block is encountered without 128 bytes in it, Xmodem pads the end of the block with an ASCII character that will not print or have any effect on the file otherwise. The extra characters only take up space to round out the block. This always happens in the last block of the file if the block has less than 128 bytes. Most other powerful micro computers store data on a disk in uniformly sized blocks. All binary files on those machines are stored that way, if Xmodem has padded the file, it doesn't matter. Those machines just ignore the excess characters at the end of the file. The problem with storing files in that way is almost every file wastes some disk space. The more files on a disk, the more space is wasted.

Disk Operating System

The Amiga is different from others. All Amiga binary files are stored with an end of file marker. Amiga Dos reads the disk, a track at a time, into a buffer and quickly discards any information it does not need. After an Xmodem file transfer, the end of the file does not match up with the end of file marker. When you try to run the file, Amiga Dos assumes that the file is something other than object code and sends an error message to the user.

Xmodem pads all uneven last blocks with filler characters during a file transfer, but it only causes a real problem with binary files. AmigaBasic, Graphic, Music, AbasiC and Text files seem unaffected. Although AbasiC files will give an Illegal Character error when loaded, this situation is easily corrected by reentering the line in question and resaving the file. Text files transferred with Xmodem will not load into Ed. Ed will report "File contains binary". Text files, including AbasiC and AmigaBasic can be fixed by using Xbin as described below.

There are several programs which are designed to remove the extra characters. The following is a list of some of the most well known. These and many others can be found on BBS's and commercial telecommunication networks.

```
Chop2.MSB---AmigaBasic....User must input filename and correct length.
Smartchop1--AmigaBasic....Auto-Chops to correct length.
Fixobj------C.............Automatic, doesn't need file length.
Trunc-------C.............Used when fixobj will not fix the file.
Xbin--------C.............For text files, doesn't need file length.

Usage:
Chop2.MSB  = Run from AmigaBasic & follow prompts.
Smartchop1 = Also runs from AmigaBasic with prompts.
Fixobj     = Drive:Dir/fixobj Drive:Dir/Filename Drive:Dir/Filename
Trunc      = Drive:Dir/Trunc Drive:Dir/filename Drive:Dir/Filename Length
Xbin       = Drive:Dir/Xbin Drive:Dir/Filename Drive:Dir/Filename
```

In using any of the above, the second filename can only be the same as the first filename if the Drive: or Dir/ paths are different. At first you need to use a program written in basic, such as Chop2.MSb or Smartchop1. Since AmigaBasic programs will run after being downloaded, they can be used to remove the Xmodem padding from Fixobj, Trunc, and Xbin which have the advantage of being faster and readily available from CLI.

In the early days of Amiga file transfers, most BBS's listed the correct file length in the file description if the length was different from the one listed in the catalog. Things have gotten a lot better. Now we have automatic file choppers such as Smartchop1, Fixobj and Xbin. These do not require the user to know the correct file length.

Smart terminal programs, such as Starterm and Wombat, chop the extra characters off the end of the file while the file is being transferred. Some files, such as Arc and Fixobj, are being prepadded so they will run without being fixed. Although prepadding is good in some cases, I don't recommend its widespread use. Why waste the disk space?

Although Hayes Verification Protocol (Smartcom) does not have the same problem, using it will not necessarily be a good solution. Downloading a file using HVP takes more time than Xmodem, which will make the call more expensive.

To make matters worse, if a file was uploaded using Xmodem and you download it using HVP, it will not work and the automatic chopper programs can not fix it. In that situation you must use a non-automatic chopper, such as Trunc, so you would need to know the correct length of the file.

If the correct length of the file is unknown, there is a simple way to determine it. At the CLI prompt enter:

`Type Drive:Dir/Filename Opt H` and press return.

Now wait until the numbers stop scrolling. At the end of the file, notice a number of occurrences of the hex number 1A. These are the extra characters that Xmodem has added to the file. Count the 1A's (Each 1A = 1). List the directory which contains the file to fix. Write down the length and subtract the number of 1A's. The new number you get is the one to use for Trunc.

The Best Solution of all is Arc

A file archive utility called Arc, which takes a file and shrinks it, can also compress files and combine them into one file with a length less than the sum of its parts. An Arc'ed file is not affected by Xmodem file transfers. This not only solves the Xmodem problem, but also cuts down on the time required to transfer a file by modem since the file is shorter. All related files are contained in one archive, which keeps files organized for easy retrieval.

•AC•

The ACO Project
Graphic Teleconferencing on the Amiga

Through Amiga Graphics, Electronic Confrence Members can now confront each other Face to Face.

By Stephen R. Pietrowicz
People Link: CBM*STEVE
UUCP: ...!ihnp4!pur-ee!gould!houligan!srp

One of the popular activities the national computer networks have to offer is online conferencing. Discussions in conferences vary greatly. Just by sitting at your computer and typing messages, you can play a game of trivia, talk to authorities on almost any subject, or even meet new friends.

Some of the people that belong to People Link's Amiga Zone club have decided that it is time for a change. They want to see more than text scrolling by when they conference. After all, the Amiga is a great graphics machine; why not use the Amiga's graphics capabilities in conferences?

The ACO (Amiga COnference) project team is working toward that goal. During conferences, people participating in the conferences "see" other conference attendees. They will see faces that the people in conference have designed themselves.

The Beginning

Back at the beginning of the summer of 1986, John Foust and I were discussing various ideas that we could implement on the Amiga. We talked about VMCO, a program that had been developed for the Apple Macintosh special interest group on CompuServe. VMCO allows the user to design faces and use them on the screen of the Macintosh during conferences to see other people's faces.

There was a rumor floating around that someone was doing a port of VMCO for the Amiga. Since no one seemed to know when it was going to be done, or even who was supposed to be writing the program, we decided to take the bull by the horns (or the Amiga by the mouse?) and do the project ourselves.

After discussing certain aspects of the project between ourselves in conferences and private mail messages, we presented the idea to the Chairman of the Commodore Club, Harv Laser. (The Amiga Zone on People Link was formerly part of the Commodore Club). Harv was very enthusiastic about the idea and offered several suggestions for the project.

After mapping out the way the program might work, we decided to go to the Amiga owners on People Link for their opinions about the project. They brought up many valuable ideas, pointed out problems, and suggested improvements. Gary Sarff, People Link ID 'GSARFF', was very helpful in creating the message packet scheme described below.

Someone suggested that we write ACO to be compatible with VMCO so we'd be able to conference with Macintosh and IBM owners. Recently, someone in the IBM club on People Link created something similar to VMCO.

There are several reasons why we decided against this. First, and most importantly, by forcing ourselves to write a Mac compatible interface, we would have to cripple the Amiga's graphics capabilities. Since the Mac only works in black and white, we'd be forced to use only black and white. Not a very appealing thought!

Eventually, we'd like to expand ACO beyond what VMCO has to offer, and expansion would mean more incompatibilities. We decided to to write our own graphics conference program, and write the specifications for it ourselves.

After the discussion died down, we decided it was time to start asking for volunteers. Sarff decided to work on the library archiver and the library routines. John Hoffman, People Link ID 'JRH', volunteered to do the graphics interface. I decided to work on the scanning routines necessary to decode packets and to integrate the different parts of ACO.

Screen layout

The ACO display shows a row of empty chairs across the top of the screen and another row of empty chairs across the bottom of the screen. Text is displayed between the two rows of chairs. As people gather for the conference, the chairs are replaced with the attendees' faces. Currently the conference screen holds room for 20 people. More people can be in conference, but their faces will not be displayed.

During conferences, text is displayed as people send messages to one another. As the text is being written to the screen, the face of that person is highlighted and kept highlighted until the text message appears on the screen. This way conference attendees will be able to keep track of who is who a little more easily.

Faces

Each person chooses one face to use during the conference from the set of faces that they designed. To change "expressions" while online, the user sends a command to ACO telling it which face to use. If the user doesn't have time to design faces or doesn't own a program that will let him draw his own set, ACO will use a set of default faces for them.

Faces for ACO are designed using DPaint. The ACO help screen used to help the user design their set of faces is loaded into the paint program. Faces are drawn using colors from the default palette and are saved as IFF brushes. These brushes are then bundled into a library format made especially for ACO and sent to a central moderator.

The moderator consolidates all the library archives into one large library for people to download from People Link's library. The current limit on the number of faces that can be used by each person while online is 5. The library archiver was written to handle more than that for future expansion, but at this time ACO allows only 5 faces.

ACO is designed to handle multiple libraries in multiple directories to avoid unmanageable libraries. After all, who wants to download a 300K file and pray for no errors? This will allow ACO to look into any directories the user specifies without defaulting to a certain drive. This way someone with a second drive or a hard disk can take advantage of it.

The ACO conference screen is designed using any IFF compatible paint program. The default screen supplied with the ACO package can be changed by anyone to show any setting. The current conference table screen could be changed to look like a cookout with everyone sitting around the campfire, a wall with talking pictures (the faces), or anything else you can think of. I suspect once ACO conferencing becomes popular, we are going to see many artists designing new ACO screens.

How does this work? The ACO package will contain a library archiver to bundle the faces you design, a few IFF pictures to help you design the faces correctly, and the ACO program itself.

ACO uses as it's base program a very popular public domain terminal program called "Comm". Dan James, People Link ID 'DJJAMES', another assistant SYSOP on People Link's Amiga Zone, was kind enough to let us use Comm as the basis for our work. So, really, you're not just getting the conferencing software, but you're also getting the rest of the features of Comm built in. Features of Comm include: macro keys, phone library, Xmodem uploading and downloading, automatic file chopping, and more. This adds the extra benefit of being ready to go into ACO conference whenever you want, without having to log off, switch programs, and dial in again.

Sample session

Upon entering the conference, the new person in the conference sends a string of characters to announce that they have entered. Each ACO participant responds by sending a message to that person to say that they are also in the conference. ACO responds automatically; the responding user doesn't have to do anything to tell the new user they are in ACO. If there is room in the conference, the new user is added to an available chair. Otherwise, the new user must wait until a new chair is freed. He can still watch the conference and participate, but the user's face will not be shown.

After the user has been established in the conference, they have several options. They can continue to send messages as in a normal conference, or they can change the face they are using on screen. By sending a command to ACO to change their face, a user can show what mood they are in at that moment. The face change will be shown to all other ACO users.

When the user wants to quit the program, ACO sends a command to free the chair that they were using. A new user can then request a chair.

The future of ACO

As of this writing, ACO is still in a limited testing phase. Gary Sarff, John Hoffman, and I have been working very hard to make ACO a reality. We constantly consult each other about ideas we have about ACO, and we are designing it to be flexible for future expansion. Possibilities for future enhancements to ACO include:

- Speech synthesis - allowing the user to hear the conversations as well as see the user's faces.

- Scrolling window - allowing the user to view lines that have already left the screen, to read something that might have gone by too quickly.

- Graphics - allowing the user to draw on the screen, sending the image to everyone else in the online conference.

By the time you see this, the first release of ACO should be available on People Link's Amiga Zone. Get your mouse in hand, draw up some faces, and join us in our weekly conferences! We'll SEE you there!

Stephen Pietrowicz works in the OSI Network Systems group of Gould Electronics, Computer Systems Division in Fort Lauderdale, FL. He is also an assistant sysop in People Link's Amiga Zone.

•AC•

Flight Simulator II
A Cross-Country Tutorial

by John Rafferty

(This article is based on ***THE FLIGHT SIMULATOR BOOK***, by John Rafferty, which is available from *En Route Books*.)

The new Amiga version of ***Flight Simulator II*** is an absolute joy. If you're ready for some truly authentic aviation, FS II on an Amiga is a pretty tough act to beat. The fighter-ace game and the biz-jet options are for casual fun, but the retractable-gear Cessna Skylane option is the choice for realistic flight. The more you know about aviation, the more impressed with it you'll be.

In brief, the program provides about 120 airports with accurate locations and runway layouts, provides scores of equally authentic VORs, radio beacons and other aids to navigation, and puts you in a sophisticated simulated aircraft in which you can fly and navigate using the same equipment and techniques as are used by airline captains on a daily basis. The whole environment is so authentic, in fact, that I routinely fly and navigate the simulator using my own aeronautical charts instead of the maps that come with the program.

Of course, this realism can be less a blessing than a curse if you don't fly realistically. If your takeoff and climb are sloppy, for example, you'll have trouble attaining straight and level flight. If you don't set up properly for the approach, your landings will tend to be fatal for all aboard. And if you're inclined to just ignore some of those more mysterious gadgets on the panel, be aware that you'll be missing out on much of the real glory of contemporary flight. The simulator responds just like the real thing, and much of the fun comes from handling it accordingly.

Learn a few basic procedures, however, and you'll be on your way to the true satisfaction and exhiliaration of modern flying--without the hazards of driving to an airport.

PREFLIGHT BRIEFING

Climb into the cockpit, take the left-hand seat, and let's take a typical commuter hop departing on Danbury Municipal's Runway 17 for JFK International. We'll stick pretty much to the basics here, but we'll execute the flight in a professional manner, using the Cessna's avionics to navigate authentically en route. We'll then encounter adverse weather as we approach Kennedy, at which time you'll file for an IFR clearance and be cleared for an instrument approach. Our flight path is shown on the accompanying chart.

COCKPIT ORIENTATION

Locating at Danbury. Boot the program, from the Nav menu click on Position set, then enter the following coordinates for the aircraft:

NORTH: 17361
EAST: 21119
ALT: 0

Now click on the Nav menu and select Map Display, to view our position on the ramp at Danbury. Then take a look at your panel.

Airspeed. The airspeed indicator, at top left, reads knots (nautical miles per hour). The Cessna cruises nicely at 130 knots, and lands comfortably at 70 with 10 degree flaps.

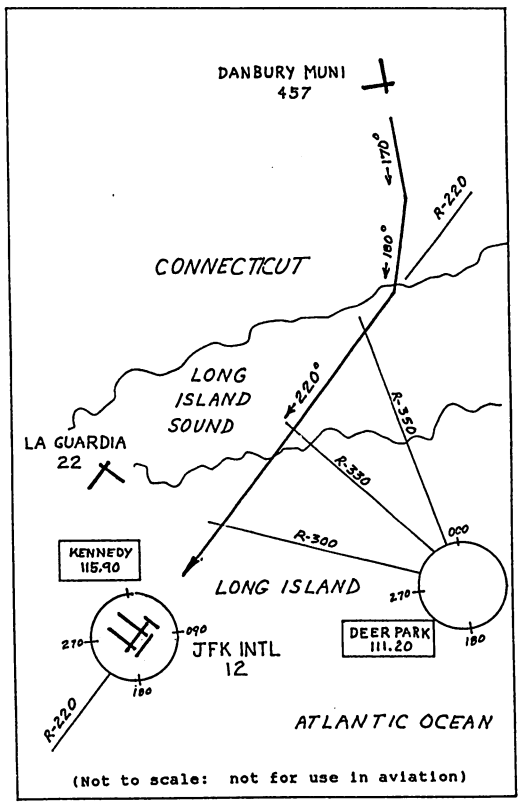
Altimeter. The altimeter, third from the left on top, is a barometer calibrated in feet. It indicates feet above sea level, not above the ground. We'll cruise at 2000 feet for the present flight--big hand on zero, small hand on 2.

Vertical Speed Indicator. The vertical speed indicator, below the altimeter, shows the rate at which the airplane is climbing or descending, and is indispensible. If the needle points up to 5 you're climbing at 500 feet per minute (a normal rate of climb). Up to 10 means a 1000 fpm climb, and down to 5 means you're descending at 500 fpm (a normal rate of descent). When you drift from your assigned altitude the vertical speed indicator tips you off first, before the altimeter begins to respond. Monitor this instrument religiously!

Magnetic Compass. The digital magnetic compass, top right, gives the airplane's heading in degrees. There's also a directional gyro, which we'll ignore on this flight.

Using the Mouse. The mouse has two modes--cursor and yoke. Click between the two with the right button. In cursor mode, use the arrow to click on menus and requestor boxes in the usual way. In yoke mode it controls the throttle, brakes, ailerons and elevator.

Throttle and Brakes. In yoke mode, advance or retard the throttle (to change engine RPM) by holding down the left button while rolling the mouse forward or back. Apply the brakes by holding down the left button and rolling the mouse left, and release them by rolling it right.



Tachometer. The digital tachometer, upper right, gives engine RPM, which is now 650. Use 1900 RPM for cruise.

Aileron Control (Turns). In yoke mode, slide the mouse left or right to turn the airplane. In the air, this banks the wings, which causes the airplane to turn. Keep your turns very shallow on the present flight. The aileron position is shown by the horizontal "slide" indicator near the center of the panel.

Elevator Control (Nose Position). In yoke mode (in the air), slide the mouse forward or back--gradually--to lower or raise the nose. The elevator position is shown by the vertical "slide" indicator on the panel. Once you have the airplane trimmed for straight and level flight, try to leave this alone: instead, favor use of the throttle to adjust and control your altitude.

Landing Gear and Flaps. The gear control is a small box on the lower right; click on the box with the cursor to raise/lower the gear (don't do that now!). Just to its right is the flap control: click on the second dot down for 10 degree flaps, on the third dot for 20 degrees, and so on. Have flaps up for a normal takeoff, and use 10 degrees for normal approach and landing.

Nav Receivers. Nav 1 and Nav 2 are stacked just to the right of center; Nav 1 is on top. You tune these radios to VOR stations--special aviation radio transmitters on the ground. The Kennedy VOR and Deer Park VOR are shown on the chart for this flight. Tune Nav 1 to the Kennedy VOR now, by using the cursor arrow to click on the two parts of the Nav radio frequency to set it to 115.90. Then tune Nav 2 to Deer Park, 111.20.

DME: Your DME (Distance Measuring Equipment) is linked to Nav 1. The digital readout (right center) indicates the distance in nautical miles to the VOR station to which Nav 1 is tuned. Thus, we're now 54.6 miles from Kennedy.

VORs. A VOR is a radio station on the ground which, in effect, sends out 360 different radio signals, one for each degree of the compass. It's like a giant wagon wheel with 360 spokes, which are called "radials." After you tune Nav 1 or Nav 2 to a given VOR, you can then select a particular radial, and then you can use the OBI window to track that radial to or from the station. It's like flying along an invisible highway in the sky.

OBI (Omni Bearing Indicator) Windows. The OBI windows for Nav 1 and Nav 2 are just to the left of the radios. Click on the selector knob at the bottom left of the Nav 1 window now, until the bearing (radial) shown above the window is 220. You're now set to Kennedy's 220 degree radial (R-220), which is shown on the chart.

OBI Needles. Now click on the OBI selector knob for Nav 2, until the bearing (radial) shown above that window is 000 degrees. The flag will say FROM, and the vertical needle will be just about centered in the window. The centered needle means we're now right "on" the 000 degree radial from Deer Park. Check the chart, to confirm that this makes sense.

TAXI, TAKEOFF AND CLIMB

Runway Numbers: Add a zero to the end of a runway number to determine that runway's approximate compass heading. Thus, Runway 35 will have a compass heading of about 350 degrees. When Runway 35 is used in the other direction, the same strip of pavement is Runway 17 (350-180=170).

Taxiing For Takeoff. Add a little power, start moving toward the runway just ahead, and turn left to parallel that runway. Note that the compass now reads about 350 degrees, indicating that we're paralleling Runway 35. Taxi ahead to just beyond the runway threshold, turn 180 degrees to the right, line up with Runway 17's centerline, and stop.

Runway Checks. The Nav 1 OBI needle should now be toward the left side of the window, which means that the Kennedy 220-degree radial is somewhere off to our left.

- *-- Check heading (on Runway 17.*
- *-- Center ailerons and elevator.*
- *-- Check flaps up.*
- *-- Write down time of departure.*

Takeoff. Add a little power to start your takeoff roll, get the airplane lined up with the centerline, then:

- *-- Full throttle.*
- *-- At 60 knots, slight up elevator (less than one "notch").*
- *-- Relax, and let the airplane fly itself off the ground.*
- *-- Once airborne, immediately raise the landing gear.*
- *-- Promptly throttle back to 2300 RPM.*

Climb. Your initial climb will be rapid, but begin at once to reduce the rate to 500 feet per minute. As soon as you've throttled back to 2300 RPM, begin to lower the nose (ease forward on the elevator control). Keep easing forward gradually on the elevator control, to hold down the rate of climb, until the vertical slide shows the elevator down one "notch" below center. Then reduce power to 2000 RPM. End up with a 500 fpm climb at 120 knots.

Climbing Turn To Heading 180 Degrees: When you've established a 500 fpm climb, bank the wings slightly to the right and begin a gentle climbing turn to heading 180 degrees. Monitor the compass, and begin to level the wings before the heading reaches 180. Make very gentle banks, if necessary, to correct your heading to exactly 180 degrees. Now also monitor the Nav 1 needle, which should be moving toward the center of the window.

Levelling Off At 2000 Feet. At 1900 feet, throttle back to 1950 RPM. As you reach 2000 feet, throttle back to 1900 RPM. Your cruising airspeed at 1900 RPM will be 130 knots. Make very very small elevator adjustments, if necessary, to peg the vertical speed indicator needle at zero; then try to leave the elevator alone. Once trimmed for level flight, if you drift from 2000 feet you can use small changes in engine RPM to make the correction. Always stay within 50 feet of your assigned altitude.

PROCEEDING EN ROUTE

Intercepting the 220-Degree Radial. Visualize the 220-degree radial from the KENNEDY VOR by referring to the chart. When the needle is one needle-width from center, bank gently to the right to start a shallow turn, and roll out on heading 220 degrees. Ideally, you'll roll out on 220 just as the needle settles at the center position. If you maintain that heading with the needle centered, you'll be flying "along" that radial directly to the airport.

Course Adjustments: If the needle isn't centered after you roll out on the 220-degree heading, or if it drifts from center later on, make a routine adjustment to center the needle. If the needle is left of center, then the radial is off to your left: turn left 5 or 10 degrees (to 215 or 210 degrees), hold that intercept heading until the needle centers, then turn right again to 220 degrees and continue on course. If the needle is off to the right instead of the left, turn right (toward the needle) to make the correction.

Cheating. Click on the Nav menu and select Autopilot, and at the Autopilot menu click the second box on the right, to "set" the autopilot on Nav 1 ("VOR 1 LOCK"). The airplane will now turn toward whatever radial is selected on Nav 1, will fly to intercept it, and will then track that radial automatically. This is great when you get busy, but autopilots can fail just when you need them most, so be able to intercept and track a radial on your own.

Position Checks: Nav 2 is on Deer Park, which is to the left of our flightpath. From time to time click on the OBI selector knob to re-center the needle, then note the resulting bearing (radial) and consult the chart to determine our position. Also refer to the DME from time to time, for our distance to the airport. At 12 miles DME you'll want to get set up for the approach.

SETTING UP FOR AN APPROACH

The key to successful landings is getting the airplane set up for the approach well in advance. If properly set up, all you have to do to begin your descent is to ease back a bit on the throttle--and then you'll find that the airplane will virtually land by itself. Use the following steps:

1. *Reduce power to about 1350 RPM.*
2. *As the nose begins to fall off, start easing back on the elevator control to keep the nose up; keep easing back, and try to keep the vertical speed indicator needle pointing at zero.*
3. *The airspeed will begin to bleed off; at 100 knots drop 10 degree flaps.*
4. *Increase RPM gradually to 1900.*
5. *Center the elevator on the vertical "slide" indicator, or just a hair below center. You'll end up at 90 to 95 knots at 1900 RPM, and you should still be at 2000 feet. Use the throttle alone to adjust altitude, if necessary.*
6. *Lower the landing gear, then begin easing back on the elevator control as the nose falls off again. You'll end up doing 70 knots with the elevator just about one " notch" above center. (You can leave the gear up for now, so as to maintain 90 knots for a little longer, but if you do then a belly landing is guaranteed, sooner or later.)*

continued...

WEATHER AND IFR CLEARANCE

Changing Weather. A layer of low-lying clouds is now moving in below us. Click on the Environment menu, then on Clouds. Click on Level 1, then click on Tops, and enter 1500. Click on Base and enter 1300, and close the menu.

Clearance. We're still flying VFR (under Visual Flight Rules), but we're above the clouds, or "VFR on top." However, you may not fly into clouds, and therefore cannot descend into JFK, unless you have an IFR (Instrument Flight Rules) Clearance from Air Traffic Control. You would therefore now call Kennedy Approach Control to request such a clearance. The exchange, in abbreviated form, might go something like the following:

Pilot: Kennedy Approach, Cessna Three Zero Four Six Foxtrot.
ATC: Cessna Three Zero Four Six Foxtrot, Kennedy.

Pilot: Kennedy, Four Six Foxtrot, VFR on top 10 miles north on Kennedy R-220, landing Kennedy, request IFR clearance, and ah, we're instrument rated and equipped.
ATC: Cessna Three Zero Four Six Foxtrot squawk 1342 you're cleared to the Kennedy VOR on heading two two zero degrees maintain two thousand contact Kennedy Tower on one one niner point one good day.

Pilot: (after switching the transponder code to 1342 and changing the Com radio frequency to 119.1) Kennedy Tower Cessna Three Zero Four Six Foxtrot.

ATC: Cessna Three Zero Four Six Foxtrot, Kennedy, you're cleared for the VOR approach to Runway Twenty-two Right altimeter three zero point five visibility 10 wind one one zero degrees at five.

Pilot: Kennedy, Four Six Fox, understand that's the VOR approach to Twenty-two Right, thank you.

You're cleared for a published instrument approach, so in a sense you're now on your own. We'll pretend you've pulled out your approach charts, have found the VOR Approach for Runway 22R, and are now following the procedures on that chart.

APPROACH AND LANDING

Instrument Approach: Before you're 8 miles DME from Kennedy, make sure that:

- *-- you're straight and level at exactly 2000 feet;*
- *-- airspeed is 70 knots with gear down and 10 degree flaps,*
- *-- heading is 220 degrees exactly;*
- *-- Nav 1 needle is on dead center;*
- *-- autopilot is off*

When the DME readout falls to 8 miles, begin to ease back gradually on the throttle. The vertical speed indicator needle will begin to fall off, indicating a descent. In gradual, patient steps, adjust the throttle for a 500 fpm descent. Try around 1500-1600 RPM.

Use power alone to maintain a smooth 500 fpm descent, and relax. Just "fly the needles," using very gentle turns, if necessary, to keep the Nav 1 needle on dead center. We'll drop into the clouds, and when we break out below, the airport should be directly ahead. Runway 22L is along the far left edge of the field, and 22R, our destination, is just a bit left of center. When you can see the runway, forget the Nav 1 needle and adjust the glide path as required.

Adjusting a Glide: Adjust your rate of descent by means of small changes in engine RPM. Note the point on the ground at which the runway begins: if that point is slowly moving DOWN on the windshield then your present glide path will cause you to OVER-SHOOT the runway threshold, so reduce power to steepen the glide. Conversely, if the touch-down point is moving UP, then you'll touch down SHORT of the threshold, so add some power temporarily. (This runway is more than two miles long.) There are other landing techniques, but this is the basic procedure.

Touch-down: Monitor the Altimeter (JFK's elevation is 12 feet), and when you're about to touch down cut the power, raise the nose slightly so as to "flare," be sure the gear is down, and let the airplane drift gently onto the pavement. (Actually, if your descent is correct the airplane will land by itself; try it.)

After touch-down, center the controls, raise the flaps, and begin to apply the brakes.

ATC: Cessna Four Six Fox turn right next intersection contact Ground on one twenty-one point niner so long.

•AC•

A Disk Librarian in AmigaBASIC™

"Disklib.BAS will generate a single listing of all your programs which includes the name of the disk and directory where the program is located"

by John Kennan

Recently, our Amiga user group began to keep a disk library of public domain programs. It quickly became obvious that we didn't have a good way of keeping track of the contents of the library. We could print all the directories, but that is not very helpful if you are trying to track down a specific program. We needed a program to maintain an alphabetical listing of our disk library.

Rather than buy a commercial program to handle that chore, I decided to write a BASIC program to keep track of the library. If you have had your Amiga for even a short time, then you have probably accumulated a number of disks as well. If that is the case, then you probably spend at least some of your time flipping through disk directories looking for the desired program.

'Disklib.BAS' will generate a single listing of all your programs which includes the name of the disk and directory where the program is located. The listing can be printed out for a hardcopy reference or kept as a text file which can be listed or searched from the CLI.

All you need do is run the program and follow the directions. The program will prompt you to insert disks to be added to the disk library. 'DiskLib.BAS' will automatically step through all the directories and subdirectories on each disk to create a complete listing of disk files.

In doing this the program ignores commonly used directories such as 'c', 'libs', 'fonts', 'devs, etc. that are reproduced on many disks and would merely clutter up the library. In addition, it disregards '.info' icon files. When you finish creating your disk library, an option in the program allows you to alphabetize the listing by program or by directory.

The program is simple to use. Since it performs a relatively simple function, I have made no effort to use colorful screens or the mouse interface, however, the program does make use of an interesting feature of AmigaBasic. AmigaBasic gives you the power to use Rom Kernel or AmigaDOS routines through the use of the Library statement.

We will make use of this feature to use the AmigaDOS List command to read all the directories off the various disks and store them in a temporary file in the RAM disk. The file in the RAM disk is then opened from the Basic program and the individual programs are then read out of the file and stored in an array for later sorting or listing.

Whenever 'DiskLib.BAS' comes upon a new directory, it is added to a list of directories which need to be searched. Whenever it has stepped through the entire list of directories, it returns to a main menu to await your next command.

To make use of this program, we have to first set up a disk to work from. This is important for single and double drive users alike, but it is especially critical for single drive users as we want to minimize the number of disk swaps that you have to make.

This is done by creating a 'startup-sequence' file that moves the DOS commands that 'DiskLib.BAS' requires to be in RAM.

Here are the steps necessary to set up a disk. Pay close attention to the instructions below.

If you have two drives, do not perform steps 8 and 9. These are only needed for one-drive systems, they transfer DOS commands to RAM. Also, these programs and files are available in the public domain, and on the AMICUS disks, so check the catalog before taking the time to enter them by hand.

Those of you with two drives can set the program up to behave as a one drive system, as this speeds up the use of the DOS commands and saves a little wear and tear on your Workbench drive.

1. Make a copy of your Workbench disk.

2. Using the Workbench Rename function, relabel this new disk to be named "DISKLIB".

3. Discard any programs you don't need. For example, the clock, the notepad, demos, etc., are unnecessary.

4. Copy the AmigaBasic program itself from the Extras disk to the DISKLIB disk.

5. Copy the 'Dos.bmap' listing from the Basicdemos drawer of the Extras disk to the DISKLIB disk.

6. Reboot on the DISKLIB disk by placing it in the internal disk drive, and pressing CTRL Amiga-Amiga. Open the CLI. If you haven't opened a CLI before, you must first go to the first screen in Preferences and switch the CLI "on". Save this preference change, and then go back to the Workbench screen. Open the 'system' drawer. The CLI icon should be visible. Double click on this icon.

7. Enter:

   ```
   RUN AMIGABASIC
   ```

   at the prompt in the CLI window.

8. You should now be in AmigaBasic. Enter Listing One. Don't worry about the fact that AmigaBasic automatically capitalizes words that it interprets as Basic keywords. It doesn't matter whether the words are upper or lower case. You are using AmigaBasic's built-in editor to make an AmigaDOS 'execute' file, a series of AmigaDOS commands that will be executed automatically. This must be saved as an ASCII file. Save the listing by using the Basic command SAVE "s/startup-sequence",A. The A option on the save command is extremely important since it specifies that the program is saved as an ASCII file. S/startup-sequence is not a Basic program!

9. Enter `NEW` at the AmigaBasic "Ok" prompt, and then enter Listing Two. Again, this is an AmigaDOS 'execute' script, so save this program by typing SAVE "s/end-sequence",A. It must be saved as ASCII, or AmigaDOS will not be able to interpret it.

10. Enter NEW in the window by the AmigaBasic "Ok" prompt, and enter listing three. This is the Basic program DISKLIB.BAS. After entering it, you can save it as you would any Basic program, by entering `SAVE "DISKLIB.BAS"` at the AmigaBasic "Ok" prompt.

With any luck, you now have a working copy of DiskLib.BAS. If you have a one drive system, you need to boot on the DISKLIB disk. If you don't want to boot on the DISLIB disk you can just go to a CLI window and enter

```
EXECUTE DISKLIB:s/startup-sequence
```

at the CLI prompt. This will place all the necessary commands into the RAM disk so 'DiskLib.BAS' will be able to access them without constantly requesting you to insert the Workbench disk.

If you have a two drive system, all you need do is place 'Disklib.BAS' in the external drive, and leave the Workbench disk in the internal drive. You will be able to remove the Workbench disk after the first menu comes up.

Enter "`RUN AmigaBASIC`" in a CLI window. When the AmigaBasic screen
appears, enter

```
Load "DiskLib.BAS"
```

at the "Ok" prompt in the AmigaBasic output window, then enter "RUN" to run the program. For 'DiskLib.BAS' to work properly, AmigaBasic must be started from the CLI. For some reason, the LIBRARY commands will not work unless AmigaBasic was started from the CLI.

Once you run the program, directions will appear that should make the program self explanatory. The program allows you to choose from a variety of options. When you choose the first option, add a disk, the program will prompt you to insert a disk to be added to the catalogue. Those of you with two drives should keep the DISKLIB disk in the external drive, and place the disk to be added to the disk library into the internal drive.

Those of you with a single drive system can remove the DISKLIB disk. Just to be on the safe side, it wouldn't hurt to write-protect the disks that you are adding to the Library before you insert them in the drive. This will prevent you from accidentally writing to the wrong disk when you try to save the disk library later.

At this point the program will begin reading the disk. You will see a few numbers printed to the screen as the program counts the number of directories it has found.

When the program is finished reading a disk, it will clear the screen and redisplay the main menu. You can now add more disks, or you can work with the newly created disk library. You'll probably want to choose option 3 to put the library in alphabetical order. Be patient, as this may take a couple of minutes. Now we can print or save the library file.

I have incorporated two Save options into the program. The first Save option, number 6, saves the library to disk in a way that the program can reload the data. It is important to save the library to disk with option 6. That way you can reload the library using the load option and add disks to the already existing library.

The second Save option, option 8, saves the library as a text file. You will want to save the program as a text file if you want to be able to access it from DOS. Remember, a file saved with option 8 cannot be accessed by the Basic program and a file saved with option 6 cannot be accessed from the CLI.

There are a number of reasons for wanting to save the library in a way that you can access it from DOS. First, 'DiskLib.BAS' can only handle a limited number of disks, I would guess between 20 and 30 disks. If you want to create a library larger than this, you should be able to save two or more separate libraries on disk as text (option 8), and then use the AmigaDOS JOIN command to create one large file. This file can then be sorted or searched from the CLI.

For example, let's say you have saved the library as 'libcli.file' using option 8. If we want to search for a program named 'Browse', we could go to the CLI window and enter

```
SEARCH DISKLIB:libcli.file Browse
```

at the CLI prompt. This AmigaDOS command searches the file for any occurrence of the text word "Browse," and will print the lines that contain it. So, when I tried it with file 'libcli.file', the SEARCH program gave this result:

```
55 Browse//AMICUS_#8:Progs/
56 Browse.c//AMICUS_#8:C/
57 Browse.DOC//AMICUS_#8:Progs/
58 BrowseMenu.c//AMICUS_#8:C/
```

Now I know that I have four programs with the word "Browse" in them, and I know the disk and directories where they can be found. This is a very easy way to find any program from the CLI.

By editing the 'libcli.file' with a wordprocessor, you could add other comments to each line of the file, to describe each program. Using the SEARCH program as above, you could easily find all the programs described as "utility", "game", etc. Remember, if you use a wordprocessor, be sure to save the disk library file as an ASCII (text only) file.

A few comments on the workings of 'DiskLib.BAS' are in order. The main reason that 'DiskLib.BAS' is able to function at all, is that it is possible to access DOS commands from within Basic. Otherwise there would be no way of reading a disk directory from within a Basic program. The important commands involved follow:

```
DECLARE FUNCTION Execute& LIBRARY
```

This command tells AmigaBasic to search opened libraries for the AmigaDOS execute command. Once we can access this function, it should be possible to directly execute any AmigaDOS command from BASIC.

```
LIBRARY "Dos.library"
```

This opens the DOS library where the execute command can be found. For this command to work, it is necessary that the 'Dos.bmap' file that you transferred from the Extras disk be present on the DISKLIB disk.


```
Execute& (SADD("list>RAM:temp "Chr$(34)+
currentdir$ + Chr$(34) + Chr$(0)),0,0)
```

This is the direct equivalent of entering:

```
LIST > RAM:temp "Directory name"
```

at a CLI prompt. Essentially we have invoked the AmigaDOS 'LIST' command for the specified directory, but we have diverted the output to the RAM disk. That places the results of this directory listing in a temporary file which we can access from AmigaBasic.

Another feature of the program which deserves mention is the use of a Shell sort to sort the programs. When I ran it, the program took about two minutes to sort ten disks worth of programs. The Shell sort is a very efficient and quick way to sort things. Actually, the two minute number is not a good reflection of the amount of time the sort takes.

You'll note that in the subroutine 'shellsort1', I invoke the function UCASE$ frequently. This is so the program sorts the file alphabetically without regard to whether the program name is upper or lower case. If one didn't invoke this function, programs with lowercase names would be placed before programs with uppercase names. If we remove all the uses of the UCASE$ function, the sort takes only about one minute.

continued...

I should note that I came across the Shell sort in an article by William Tracy in the March/April 1984 issue of *Sync Magazine*. It is generally much faster than a bubble sort, and it is not that much more difficult to implement.

Another useful feature of 'DiskLib.BAS' concerns the print option. If you look at the print subroutine named 'pprint', you'll notice that the first thing it does is call another subroutine called setprt. This subroutine sends a set of control characters to the printer which cause the following to occur:

1. The printer is reset to its startup configuration.

2. The printer is set up to print 66 lines per page.

3. The printer is told to skip 12 lines at the bottom of the page. This effectively causes the printer to skip the perforation.

It is a good idea to position the paper to the beginning of a page before you invoke the print to printer option. The listing which is produced is then easily separated into individual sheets which can be placed in a folder.

Another feature in 'DiskLib.BAS' that all AmigaBasic programmers should be aware of concerns the CLEAR command. Many of you probably wonder why I use CLEAR twice in succession at the beginning of the program.

Well, apparently there is a small bug in AmigaBASIC™. If you run a program which clears a large amount of memory, you will frequently get an out of memory error if you run it a second time. Apparently clearing to a smaller memory size first allows you to clear to the larger number again.

I noticed this used in a program called *Pyramid Power* by Mike Lightstone which appeared in the October 1986 issue of *Compute Magazine*. He didn't comment on why he did it, but it certainly seems to make things run more smoothly in my programs. If you should happen to get an out of memory error message, you can always reduce the memory requirement by decreasing the 50000 in the second CLEAR statement to some smaller value (this may decrease the capacity of the library you can handle).

One last point. When you are through with 'DiskLib.BAS', exit the program through the exit menu option. The system will prompt you to insert the disk you started up with when you leave the program. You are now free to leave Basic as you normally would.

One-drive users should then go to the CLI window and enter:

```
Execute DISKLIB:s/end-sequence
```

at the CLI prompt. This will remove the files we placed in the RAM disk and restore things to pretty much to the way they were before you ran 'DiskLib.BAS'.

That completes the use of 'DiskLib.BAS'. I hope this program proves useful to all of you with large disk libraries.

'DiskLib.BAS' At A Glance

'DiskLib.BAS' is an AmigaBasic program to maintain an index of a disk library. The program can create a hardcopy or text file listing of all your programs. The listing includes all the information you need to find the location of the program on disk. To make a working copy of 'DiskLib.BAS', the program must reside on a disk named DISKLIB. To create a working disk, just follow this procedure:

1. *Copy a Workbench Disk*
2. *Delete unnecessary file such as the Clock or Demos drawer*
3. *Rename this copy to DISKLIB*
4. *Enter the following lines from the CLI*

```
Copy Extras:AmigaBasic to DISKLIB:
Copy Extras:Basicdemos/dos.bmap to DISKLIB:
Copy s/startup-sequence to DISKLIB:s
Copy s/end-sequence to DISKLIB:s
```

(note that files from the s directory must be copied from this disk)

You now have a working copy of 'DiskLib.BAS'. Just boot on the DISKLIB disk and run AmigaBASIC™ from the CLI. Then load and run disklib.bas.

A useful feature of 'DiskLib.BAS' is the ability to save a disk library as a text file. The text file can be searched from DOS. For example if the disk library is named 'LIBCLI.FILE' and you are searching for a program named Browse, just go to the CLI and enter **"SEARCH DISKLIB:LIBCLI.FILE Browse"**

AmigaDOS will return a list of every line in the file that contains the text string "Browse". You can also load the text file into ED and add a brief comment to each line, to describe the program. These comments can also be SEARCHed in the same manner. I hope you find 'DiskLib.BAS' to be a useful program.

Listing One

```
disklib:c/assign c: disklib:c
cd disklib:
loadwb
echo "This startup-sequence is intended to allow single"
echo "drive users"
echo "to use the program disklib.bas"
echo "To reestablish your original configuration"
echo "when you are done with disklib.bas, enter:"
echo ""
echo "execute disklib:s/end-sequence"
echo ""
assign l: disklib:l
assign libs: disklib:libs
assign devs: disklib:devs
IF NOT exists ram:c
makedir ram:c
endif
copy c/info ram:c
copy c/cd ram:c
copy c/LIST ram:c
copy c/assign ram:c
copy c/execute ram:c
```

```
copy c/RUN ram:c
copy disklib:dos.bmap TO ram:
newcli
assign c: ram:c
df0:c/endcli > nil:
```

Listing Two

```
; Restores everything after Disklib:s/startup-sequence has
been executed
assign c: sys:c
assign devs: sys:devs
assign l: sys:l
assign libs: sys:libs
IF exists ram:dos.bmap
DELETE ram:dos.bmap
endif
IF exists ram:c
DELETE ram:c ALL
endif
```

Listing Three

```
REM disklib.bas by John Kennan
CLEAR,20000
CLEAR,50000&,5000
OPTION BASE 1
DIM prog$(1500),dir$(400),prog%(1500)
dircheck=0:dircount=0:progcount=0
dname$=""
DECLARE FUNCTION Execute& LIBRARY
PRINT "For this to work properly, AmigaBasic must be run"
PRINT "from the CLI!
PRINT "If you have a 1 drive system,  boot on the DISKLIB:"
PRINT "disk, or go to the CLI and enter:"
PRINT "`Execute Disklib:s/startup-sequence'"
PRINT "On a 2 drive system, place Disklib: in the Ext Drive"
PRINT "and use the int drive to add disks to the library."
WHILE ((drive<>1) AND (drive<>2))
INPUT;"Are you working on a 1 or 2 drive system :";drive
WEND
CLS
IF drive=1 THEN
  CHDIR "RAM:"
 ELSE
  CHDIR "disklib:"
  LIBRARY "dos.library"
  x=Execute&(SADD("Assign c: disklib:c"+CHR$(0)), 0, 0)
  x=Execute&(SADD("Assign l: disklib:l"+CHR$(0)), 0, 0)
 x=Execute&(SADD("Assign devs: disklib:devs"+CHR$(0)), 0, 0)
 x=Execute&(SADD("Assign libs: disklib:libs"+CHR$(0)), 0, 0)
  x=Execute&(SADD("CD disklib:"+CHR$(0)), 0, 0)
  LIBRARY CLOSE
END IF
main:
  CLS
PRINT "Library contains " progcount " progs (max 1500) and"
  PRINT dircount " directories (max 400)"
  PRINT "You have " FRE(1) "bytes of free memory in Basic"
  PRINT "Choose an option"
  PRINT "1-Add a Disk"
  PRINT "2-Sort by Directory"
  PRINT "3-Sort by Program"
  PRINT "4-Print Library to Screen"
  PRINT "5-Print Library to Printer"
  PRINT "6-Save Library to Disk"
  PRINT "7-Load Library from Disk"
  PRINT "8-Save Library as a Text File"
  PRINT "9-Delete a Directory"
  PRINT "10-Exit Program"
  choice=-1
  WHILE ((choice>10) OR (choice<1))
  INPUT;"Enter option 1-10:";choice
  WEND
  ON choice GOSUB adddisk,shellsort1,shellsort2,scprint,
                  pprint,fsave,fload,textsave,dirdelete,exprog
  GOTO main
adddisk:
  PRINT "Insert Disk to be added into DF0: and press 'C'";
  PRINT "to continue or 'A' to abort"
  a$=""
  WHILE ((a$<>"C") AND (a$<>"A"))
  a$=INKEY$
  a$=UCASE$(a$)
  WEND
IF a$="A" THEN RETURN
CALL diskname(dname$)
IF dname$="" THEN
  PRINT "NO DISK IN DRIVE!!"
  GOTO adddisk
END IF
dircheck=dircheck+1:dircount=dircount+1
dir$(dircheck)=dname$+":"
flag=0
WHILE flag=0
 CALL listram(dir$(dircheck))
 OPEN "ram:temp" FOR INPUT AS #1
WHILE NOT EOF(1)
  LINE INPUT #1, temp$
  IF (INSTR(temp$,"Directory")=0 AND INSTR(temp$," Dir "))
THEN
  IF (LEFT$(temp$,4)<>"Dir ") THEN
  IF (LEFT$(temp$,7)<>"MakeDir") THEN
   IF (INSTR(temp$,"directories")=0) THEN
    dircount=dircount+1
    position=INSTR(temp$,"    ")
    newdir$=LEFT$(temp$,(position-1))
    dir$(dircount)=dir$(dircheck)+newdir$+"/"
    IF newdir$="c" THEN dircount=dircount-1
    IF newdir$="Trashcan" THEN dircount=dircount-1
    IF newdir$="l" THEN dircount=dircount-1
    IF newdir$="s" THEN dircount=dircount-1
    IF newdir$="libs" THEN dircount=dircount-1
    IF newdir$="fonts" THEN dircount=dircount-1
    IF newdir$="devs" THEN dircount=dircount-1
   END IF
   END IF
   END IF
  ELSE
   IF ((INSTR(temp$,"irectory")=0) AND
       (INSTR(temp$,"irectories")=0)) THEN
   IF ((INSTR(temp$,"ile -")=0) AND
       (INSTR(temp$,"iles -")=0)) THEN
   IF (INSTR(temp$,".info")=0) THEN
    progcount=progcount+1
    prog%(progcount)=dircheck
    position=INSTR(temp$,"    ")
    IF position THEN position=26
    prog$(progcount)=LEFT$(temp$,(position-1))
   END IF
   END IF
  END IF
  END IF
 WEND
  CLOSE #1
  KILL "ram:temp"
PRINT "dircheck=" dircheck
PRINT "dircount=" dircount
dircheck=dircheck+1
IF dircheck>dircount THEN flag=1
WEND
dircheck=dircount
  LIBRARY CLOSE
RETURN
scprint:
  n=1:m=0
  WHILE n<= progcount
  WHILE m<20
  IF n<=progcount THEN
```

continued...

```
      PRINT prog$(n) TAB(30) dir$(prog%(n))
  n=n+1:m=m+1
  WEND
  m=0
  PRINT "To Continue hit 'C' (or 'A' to abort)"
    a$=""
    WHILE ((a$<>"C") AND (a$<>"A"))
    a$=INKEY$
    a$=UCASE$(a$)
    WEND
  IF a$="A" THEN n=progcount+1
  WEND
RETURN
pprint:
  GOSUB setprt
  FOR n=1 TO progcount
  LPRINT prog$(n) TAB(30) dir$(prog%(n))
  NEXT n
RETURN
textsave:
  PRINT "If you are using a one drive system"
PRINT "Filename must include the drive specifier"
PRINT "or the disk NAME, or the file will be saved TO RAM:"
INPUT;"Input file name to be SAVED (enter 'A' to abort)";a$
  a$=UCASE$(a$)
  IF a$="A" THEN RETURN
  b$=""
  IF (drive=1) AND (INSTR(a$,CHR$(58))=0) THEN
      PRINT "You did not include the drive or disk name!!"
 INPUT;"Do you want to save to df0: or abort (S or A)?:";b$
      b$=UCASE$(b$)
      IF b$="S" THEN a$="df0:"+a$
  END IF
  IF b$="A" THEN RETURN
  OPEN a$ FOR OUTPUT AS #1
  FOR n=1 TO progcount
  PRINT #1,prog$(n) "\\" dir$(prog%(n))
  NEXT n
  CLOSE #1
  RETURN
fsave:
  PRINT "If you are using a one drive system"
PRINT "Filename must include the drive specifier (ie.
DF0:)"
PRINT "or the disk NAME, or the file will be saved TO RAM:"
INPUT;"Input file name to be SAVED (enter 'A' to abort)";a$
  a$=UCASE$(a$)
  IF a$="A" THEN RETURN
  b$=""
  IF (drive=1) AND (INSTR(a$,CHR$(58))=0) THEN
      PRINT "You did not include the drive or disk name!!"
 INPUT;"Do you want to save to df0: or abort (S or A)?:";b$
      b$=UCASE$(b$)
      IF b$="S" THEN a$="df0:"+a$
  END IF
  IF b$="A" THEN RETURN
  OPEN a$ FOR OUTPUT AS #1
  PRINT #1,dircount;progcount
  FOR n=1 TO dircount
  WRITE #1, dir$(n)
  NEXT n
  FOR n=1 TO progcount
  WRITE #1,prog$(n),prog%(n)
  NEXT n
  CLOSE #1
  RETURN
fload:
  ON ERROR GOTO 0
PRINT "Filename must include the drive specifier (ie. DF0:)"
PRINT "or the disk NAME, or it will be loaded from RAM:"
INPUT;"Input file to be LOADED (enter 'A' to abort)";a$
  a$=UCASE$(a$)
  IF a$="A" THEN RETURN
  ON ERROR GOTO diskerror
  OPEN a$ FOR INPUT AS #1
  ON ERROR GOTO 0
  INPUT #1,dircount,progcount
  FOR n=1 TO dircount
  INPUT #1, dir$(n)
  NEXT n
  FOR n=1 TO progcount
  INPUT #1,prog$(n),prog%(n)
  NEXT n
  dircheck=dircount
  CLOSE #1
  RETURN
diskerror:
  PRINT "FILE NOT FOUND!!":PRINT
  RESUME fload
shellsort1:
  PRINT "Sorting...."
  s=2^INT(LOG(progcount)/LOG(2))
  IF s<(progcount/2) THEN s=s*2
 startsort:
  s=INT(s/2)
  IF s=0 THEN RETURN
  FOR t=1 TO progcount-s
    y=t
   L1:
    w=y+s
  IF UCASE$(dir$(prog%(y)))<=UCASE$(dir$(prog%(w))) THEN
   GOTO Skip1
      SWAP prog%(y),prog%(w)
      SWAP prog$(y),prog$(w)
      y=y-s
    IF y>0 THEN GOTO L1:
  Skip1:
   NEXT t
   GOTO startsort
shellsort2:
  PRINT "Sorting..."
  s=2^INT(LOG(progcount)/LOG(2))
  IF s<(progcount/2) THEN s=s*2
 startsort2:
  s=INT(s/2)
  IF s=0 THEN RETURN
  FOR t=1 TO progcount-s
    y=t
  L2:
    w=y+s
    IF UCASE$(prog$(y))<=UCASE$(prog$(w)) THEN GOTO Skip2
      SWAP prog%(y),prog%(w)
      SWAP prog$(y),prog$(w)
      y=y-s
   IF y>0 THEN GOTO L2:
  Skip2:
   NEXT t
   GOTO startsort2
SUB listram(currentdir$) STATIC
  LIBRARY "dos.library"
  x=Execute&(SADD("list >RAM:temp "+CHR$(34)
          +currentdir$+CHR$(34)+CHR$(0)), 0, 0)
  LIBRARY CLOSE
END SUB
SUB diskname (dname$) STATIC
  LIBRARY "dos.library"
  x=Execute&(SADD("info >RAM:temp"+CHR$(0)), 0, 0)
  OPEN "RAM:temp" FOR INPUT AS #1
  position=0
  WHILE position=0
    LINE INPUT #1,a$
    position=INSTR(a$,"Name")
  WEND
string=0
  WHILE string=0
    LINE INPUT #1,a$
    string=INSTR(a$,"DF0:")
  WEND
  IF INSTR(a$,"No disk present")<>0 THEN
  dname$=""
  ELSE
  dname$=RIGHT$(a$,LEN(a$)-position+1)
  END IF
```

```
  CLOSE #1
  KILL "RAM:temp"
  LIBRARY CLOSE
END SUB
setprt:
OPEN "prt:" FOR OUTPUT AS #1
PRINT #1,CHR$(27);"c";
PRINT #1,CHR$(27);"[66t";
PRINT #1,CHR$(27);"[12q";
CLOSE #1
RETURN
dirdelete:
INPUT;"Input Dir to delete (enter 'A' to abort)";dirdel$
  dirdel$=UCASE$(dirdel$)
  IF dirdel$="A" THEN RETURN
  zz=dircount
  FOR n=zz TO 1 STEP -1
  IF INSTR(dir$(n),dirdel$)<>0 THEN GOSUB ddelete
  NEXT n
  dircheck=dircount
  RETURN
ddelete:
  PRINT "Delete" dir$(n) "?(Y or N):":INPUT a$
  a$=UCASE$(a$)
  IF a$<>"Y" THEN RETURN
  FOR m=n TO dircount-1
  dir$(m)=dir$(m+1)
  NEXT m
  dircount=dircount-1
  m=1
  WHILE m<=progcount
  IF prog%(m)=n THEN

    FOR z=m TO progcount-1
    prog%(z)=prog%(z+1)
    prog$(z)=prog$(z+1)
    NEXT z
    progcount=progcount-1
  END IF
  IF prog%(m)>n THEN prog%(m)=prog%(m)-1
  IF prog%(m)<>n THEN m=m+1
  WEND
RETURN
exprog:
  LIBRARY CLOSE
  CLOSE #1
  CHDIR "Disklib:"
  IF drive=2 THEN
    LIBRARY "dos.library"
    x=Execute&(SADD("Assign c: sys:c"+CHR$(0)), 0, 0)
    x=Execute&(SADD("Assign l: sys:l"+CHR$(0)), 0, 0)
    x=Execute&(SADD("Assign devs: sys:devs"+CHR$(0)), 0, 0)
    x=Execute&(SADD("Assign libs: sys:libs"+CHR$(0)), 0, 0)
    LIBRARY CLOSE
  ELSE
PRINT "To restore system as before you ran the program"
PRINT "Go to CLI and enter `Execute disklib:s/end-
sequence'"
  END IF
END
```

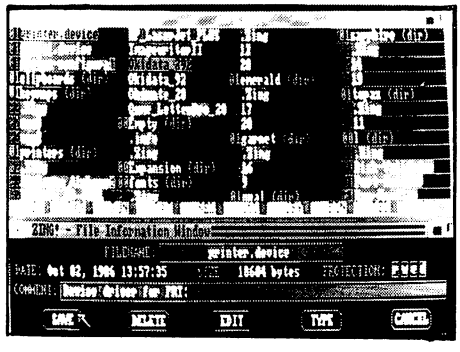
•AC•

ZING!™ is an exciting new software package which provides a fast and powerful interface between the user and the computer. You'll be amazed at the power packaged in this little disk; yet it's so simple to use, you'll wonder why no one else has thought of it before. You no longer have to resort to typing cryptic commands through CLI. **ZING!** uses Intuition™ which provides you with easy window, icon, menu and mouse controlled features.

PACKED WITH POWER!

"Sending Files to the Print Spooler"

"ZING! Hot Keys"



"Modifying a File"

FEATURES

- Install Disks
- Copy Disks
- Relabel Disks
- Rename files
- Display a file tree
- Select files by mouse
- Select files by pattern
- Select files by time
- Set file protection
- Delete files
- Move files
- Create directories
- Change directories
- Piping of file names
- Built-in screen saver (dimmer)
- Fast Sort directory display
- Start editor with no typing
- Hot Keys
- Merge files
- Copy files
- Support full multi-tasking
- SPOOL files to the printer
- Save screens to the printer
- Save screens to IFF files
- Reassign function keys
- Format Disks
- Fancy file browser
- Monitor system tasks
- Set file comments
- Run programs from **ZING!**
- Show status of devices
- Assign internal symbols
- Show available memory
- Copy all or PARTIAL file trees

ZING! offers these and hundreds of other capabilities without preventing you from running other applications simultaneously. Order **ZING!** and transform your mild mannered CLI into the fastest and most powerful computer interface ever conceived! It's available now for the **special introductory price** of **$79.95** plus $3.00 shipping and handling.

MERIDIAN™ SOFTWARE INC.
P.O. Box 890408
Houston, TX. 77289-0408

(713) 488-2144
Credit Cards and
Dealer Inquiries Welcome!

ZING! is a trademark of MERIDIAN SOFTWARE, INC.
AMIGA is a registered trademark of Commodore-AMIGA, Inc.

CREATING and USING AMIGA WORKBENCH ICONS

"... easily personalize Workbench by designing and using your own icons."

By Celeste Hansel

Icons serve as a window into the workings of Amiga DOS. In the Workbench environment, icons take the place of file names, and prove that "one picture is worth a thousand words".

Some of the icons that come with Workbench are drawer pictures for directories, and disk pictures for disks. Some icons are customized to the application, such as the clock, notepad, and calculator.

Creating new custom icons is easy. You don't have to be an artist to make them. Using different icons for different files can help make file handling in Workbench easier.

If you have many files, it may take a while finding a certain file if all the icons are of the generic type. If you had letter icons for letters, note icons for notes, and list icons for lists, finding the right one would be a snap. This article will show you how to easily personalize Workbench by designing and using your own icons.

An icon is a file with the ".info" extension. This file contains all the graphical information about that icon, and the icon type, as well as information needed when the icon is displayed, such as the position of the icon on the screen, or the size of the window associated with a drawer.

When you create a new icon, you replace the graphic information in the appropriate type of file and rename it. If you are going to replace an existing icon, you save the new icon with the file name of the icon file to be replaced. If you are creating a new icon for an existing directory or file, you save the new icon with the name of the directory or file.

The Icon Editor adds the ".info" part of the name, called an extension. The icon file will be saved on the current disk, in the current drawer, unless you specify the path. Paths are a fully descriptive file names, which specify the name of the disk, followed by a colon, and then any drawer (directory) names, separated by slashes, (if there is more than one,) and end with the name of the file itself.

If you don't know the path of the icon, you may save it in the System directory on Workbench and then copy the new icon file to the old icon file. This is done by dragging the new icon to the window where the icon's directory or file resides, or using the COPY command in CLI. Either way, you need to know the file name of the icon you are replacing. In most cases, it is the name of the icon's corresponding directory, file, etc. with the ".info" extension.

To edit existing icons or create new ones, you use the Icon Editor provided on your Workbench disk. To get into the Icon Editor, open the System drawer in the Workbench window, then open the IconEd icon.

The Icon Editor will appear with a requester in front that lists the different types of icons. The icon type DISK is the root of a disk, DRAW is for a directory on the disk, TOOL is for a directly runable program, PROJECT is for a data file, GARBAGE is for the trash can directory. One that is not listed in the user manual, but is in the ROM Kernel Manual, is KICK for a non-DOS disk.

Once you get past the icon type requester, you are in the Icon Editor. The sketchpad for drawing your new icon is on the left side. To the right are nine extra pages. These pages can hold up to nine separate icons or nine versions of the same icon.

If the type of icon you want to create is a directly runable program you are ready to start, because the IconEd icon is loaded, and it is of the program type. If you want to create another type of icon you must first load an icon of that type.

To do this, pick Load Data from the Disk menu. In the requester that appears, click in the box under "Enter Icon Name". Erase the name that is currently in the box, by using the Delete or Back Space key at the top right corner of the keyboard, and enter the path and name of an icon file of the type you want.

Figure 1 shows a window with different examples of icons. A good source of icon figures can be found in cross stitch and needlepoint books. Icons can also be charted on graph paper using two vertical squares for one pixel. It is best to make icons about half the size of the sketchpad. If they are the full size of the sketchpad, they will take up a lot of space in your window.

The currently selected page is shown on the sketchpad, and bordered in black in the boxes to the right of the sketchpad. The rest of the pages have white borders. To place a particular page on the sketchpad, select that page and it will appear on the sketchpad to the left.

When you want to make changes to an icon you are working with, but aren't sure how it will turn out, you can either select Snapshot Frame or copy the current icon to a new page and play with it. If you don't like the changes you have made, you may go back to the previous icon.

To copy the current icon to a new page, select a new page, then from the Copy menu choose Copy From Frame and the number corresponding to the page of the icon being copied. You will now have two identical icons, one to work with and one as a backup.

To save an icon, pick Save Data from the Disk Menu. In the requester that appears, select the box under "Enter Icon Name", erase what is currently in the box, and type the path (if known) and name of the icon file your new icon will replace. If you want to save the entire sketchpad area select Save Full Image.

For instance, if you want to create an icon for a directory, you need a DRAW type icon, and you could load the icon whose name and path is "df0:System". If you wanted to create a data file icon, the PROJECT type, and had at one time created a data file using Notepad and named it MY_LIST, then you could load df0:Utilities/MY_LIST. Table 1 gives a brief description of the Icon Editor Menus.

To clear the IconEd sketchpad, pick 'Clear This Frame' from the Misc Menu. This will give you a blank sketchpad approximately 80 pixels across and 42 pixels down. Pixels appear rectangular because the program is compensating to make the enlarged image icon look normal. When working on a new icon, plan each pixel to be about twice as tall as wide.

You can place a single pixel on the sketchpad by pointing the arrow and pressing the left mouse button once, or draw a continuous stream of color by holding down the left mouse button and moving the arrow across the sketchpad. The colors can be changed by picking different colors from the Color Menu. These colors are your Workbench colors and can be changed using Preferences. To erase, simply choose the background color and draw over the mistake.

If you want to save space and your icon doesn't take up the entire sketchpad, select 'Frame And Save.' Move the arrow to the upper left corner of the section on the image on the sketchpad you want to save, press the left mouse button once. This should make a stretchable box. Move the arrow to the lower right corner of the portion to be saved. You will see the frame grow around your icon. Press the left mouse button when the frame is positioned correctly, and the smaller icon will be saved.

If the path was not given when the new icon was named, you must copy the new icon file over the old icon file. You can do this using CLI and the COPY command, or by dragging your new icon to the window containing the old icon. The new icon appears in the system drawer after you save it, behind the IconEd icon. To see it, close the System drawer, and and then open it again.

With a little imagination and experimenting with the Icon Editor, you can draw great looking customized icons. Using icons you have created can make Workbench easier to use, files easier to find, and make your friend the Amiga even friendlier.

Figure One

ICON EDITOR MENUS

Color
This menu chooses Blue, White, Black and Gold, based on your Preferences colors.

Copy
'*Undo Frame*' deletes the currently selected frame and replaces it with the frame saved during the last snapshot frame. If no snapshot frame was done this session, the currently selected frame is replaced with a blank frame.

'*Snapshot Frame*' saves a copy of the currently selected frame during the current session. Snapshot only saves the most recent snapshot. Be careful!

'*From Frame*' copies from frame number highlighted in the submenu to the currently selected frame. The icon that was in the currently selected frame is lost if not saved or snapshot. Note: the frames are numbered from right to left, top to bottom.

'*Merge With*' merges the currently selected frame with the frame frame number highlighted in the submenu. If a pixel from each frame overlaps the same position, the color that appears is determined by certain rules found in the Merging Frames section of Appendix E in "Introduction to Amiga".

Move
'*In Frame*' moves the icon up, down, right, or left inside the currently selected frame.

'*Exchange With*' exchanges the currently selected frame with the frame number highlighted in the submenu.

Text
'*Write Into*' allows you to include text in an icon. The text frame should be fairly short, less than six letters. The adding text to an icon section of Appendix E in "Introduction to Amiga" gives more than enough information on this topic.

Disk
'*Save Data*' saves the currently selected icon, that is, it saves the icon image on the sketchpad.

'*Load Data*' loads an icon file into the icon editor. You need to specify the path and name of the icon.

Misc
'*Clear*' erases the image of the currently selected icon.

Frame
'*Flood Fill*' fills an area with color. To do this, select 'Flood Fill', then point to the area to be colored and press the left mouse button. The area will be filled with the currently selected color from the color menu.

'*Set Bottom*': If 0 is selected, there will be no space between border, the bottom of the icon and the icon name. If 1 is selected, there will be one space between the bottom of the icon and the icon name.

Highlight
'*Inverse*': If 'Inverse' is chosen, when an icon is selected, the entire icon frame will be shown in inverse video, that is, the colors change.

'*Backfill*': If backfill is chosen when an icon is selected on the Workbench, it will be shown in inverse video, except for the background which stays the same.

•AC•

Roomers

"Something is afoot inside Commodore, and something new and exciting is sure to be shown at the *Consumer Electronics Show*"

By The Bandito

Move over, Amigo. Sorry, Mr. AMICUS, you don't have what it takes to get the best rumors in the Valley.

Sockless Amigas
Did you know the Amiga is a regular on *Miami Vice*? You can spot it at headquarters. It might be painted black, so look carefully. It has been spotted running Preferences. The Amiga also tagged along with the Vice co-stars when they hosted *Friday Night Videos.*

I also heard star musician *Frank Zappa* bought several Amigas, along with the Soundscape software from *Mimetics.* In an interview, actress *Amy Irving* said her son loves to play *Marble Madness* at home. Given that her husband is *Steve Spielberg*, this means the tot either has an Amiga, a Commodore 64, or the real stand-up game...

Ho, ho ho
December saw several new rumors fly across the country. The best is the *Reindeer' machine*, reported in Infoworld's gossip column, a 68020 machine with a 80386 card for IBM compatibility. Another popular rumor talked about the Amiga 8500.

My best sources say this '8500' is an alcohol-induced joke, promulgated by partying Commodore West Chester employees. Variations of the rumor were to be spread in different cities after the show, just to track the flow of information. One such variation was the 'TeraBaud' connector, a mythical hyper-fast device interface.

But don't take this denunciation too seriously. Something is afoot inside Commodore, and something new and exciting is sure to be shown at the *Consumer Electronics Show* in January, in Las Vegas. The techies at Commodore West Chester are working on something top-secret. Even macadamia nuts won't budge their lips. Other connections claim all new Commodore products will be developed in Germany.

The sure bet says CBM will show the PC-10, their IBM clone, in two models, with one and two disk drives, and perhaps an AT clone. Zzzzzz.

Others say they will show the Amiga 500 and Amiga 2500 machines to select dealers, behind closed doors. Someone else think it will be an official announcement of the new machines. Another source ventures that Commodore will have the Amiga 500 and 2500 in production and shipping much earlier than anyone expects, such as the March-April time frame.

Another HAM editor
Keep an eye peeled for ***Prism***, a new HAM editor from *Genesys Technologies.* This means there are two hold-and-modify editors in the Amiga market, Prism and Digi-Paint. The beta test versions of Prism look more like Deluxe Paint than Digi-Paint. The palette requester is beautiful in itself. Six squares of colors are shown, each contains a different smooth range of colors and shades, and colors are chosen by clicking on the color you want.

Defender love scene
I heard about alternate endings in the ***Defender of the Crown*** game. In case you have not seen it, the final scene is a love scene, where the handsome knight draws closer to the gorgeous damsel, by the light of a fireplace.

According to an insider, the scene formerly required certain mouse actions to coax the fair woman, who reacted accordingly. After consulting with game designers and project management, the programmer supposedly destroyed all copies of the source code to that scene.

Defender of the Crown might just get my award for most anti-woman game of the year. Women are treated as property in this game. Their perky little faces show up as trophies on the scoring screen. "Ugh. Me Galahad, You Jane."

Mac emulator card
A company named *Data Pacific* in Denver, Colorado may be readying a Macintosh emulator card for the Amiga. They presently produce one for the Atari ST. Someone thought it might be ready this spring, but they are testing the waters, to see if such a product is possible, and if there is a market for it. Call (303) 733-8158 to tell them so.

Kahn they do it?

Borland president *Philippe Kahn* stopped by the *Compuserve Amiga Forum* to confirm rumors that they are preparing ***Amiga Turbo Pascal.*** According to Kahn's message, "As all good software takes time, I cannot yet announce a release date. But we will have the product, provided the 'sky doesn't fall on our heads,' as the ancient Gauls used to say! I think that it will be an exciting product for a great machine! -Philippe-". Straight from the horse's mouth?

CBM 800 number

I heard why Commodore stopped their toll-free 800 support number. CBM president *Thomas Rattigan* overheard his son talking to tech support for more than 30 minutes, and he pulled the plug the next day...

New Transformer

According to rumors coming from Commodore West Chester, a new version of ***Transformer*** has been ready since June 1986, but something is stalling it from coming to market. Commodore was unhappy with the initial delays in delivery of the first version of the Transformer, and hesitate to pay Simile for a second revision. This improved version is supposedly faster, works under AmigaDOS 1.2, and takes advantage of memory expansion.

Grand Prix, Jet, Gunship

Look out for a new game from *Electronic Arts*, called ***Grand Prix.*** This is a car racing simulation. You can choose the race track from a menu, and each car and the course itself is drawn with what looks like 3-D animation of solid, shaded objects. Even the weather is simulated, and the rear-view mirrors show you the view behind! This program is written entirely in assembly language, for speed.

A loose-lipped *Electronic Arts* insider claims the game from former Amiga programmer *RJ Mical* is on schedule. The game encompasses a 3-D space. The player viewpoint is similar to the view from the ball in a pinball machine, and moves within the confines of the animated space, bouncing around and interacting with the walls.

EA is also working on new versions of ***Deluxe Video*** , according to one insider.

SubLogic was reportedly caught off guard when the first production run of ***Amiga Flight Simulator II*** sold out, causing them to take a good look at the Amiga market. They are working on Amiga Jet, expect it in mid-spring.

Another microcomputer flight simulator company, *Microprose*, is porting their game ***Gunship*** to the Amiga, scheduled for release in the same time frame.

Expect a new version of *Micro-Systems Software's* ***Analyze*** spreadsheet, version 2.0. It has new graphics functions, a macro language, direct read and write of *Lotus .WKS files*, and the ability to import and export data to and from Scribble!.

•AC•

Amazing Computing™

If you are reading *Amazing Computing™* for the first time, you have not seen *Amazing Computing™*. Look what you have missed!

Volume 1 Number 1 Premiere February 1986

Super Spheres By Kelly Kauffman An ABasic Graphics program
Date Virus By John Foust There is a disease that may attack your Amiga
EZ-Term by Kelly Kauffman An ABasic Terminal program
Miga Mania by Perry Kivolowitz Programming fixes and mouse care
Inside CLI by George Musser a guided insight into the AmigaDos™
CLI Summary by George Musser Jr. A removable list of CLI commands
AmigaForum by Bela Lubkin A quick trip through Compuserve's Amiga SIG
Commodore Amiga Development Program by Don Hicks What to ask and where to go to be a developer
Amiga Products A listing of present and expected products.

Volume 1 Number 2 March 1986

Electronic Arts Comes Through A look at the new software from EA
Inside CLI: part two by George Musser George continues his investigation of CLI and ED
A Summary of ED Commands
Live! by Rich Miner A review of the Beta version of the Live! frame grabber
Online and the CTS Fabite 2424 ADH Modem by John Foust
Amiga Products
Superterm V 1.0 By Kelly Kauffman A terminal program written in Amiga Basic
A Workbench "More" Program by Rick Wirch
Amiga BBS numbers

Volume 1 Number 3 April 1986

Analyze! a review by Ernest Viverios
Reviews of Racter, Baratacca and Mindshadow
Forth! The first of our on going tutorial
Deluxe Draw!! by Rich Wirch An Amiga Basic program for the artist in us all.
Amiga Basic, A beginners tutorial The start of our tutorial of the most active Amiga language.
Inside CLI: part 3 by George Musser George gives us PIPE

Volume 1 Number 4 May 1986

SkyFox and Articfox Reviewed
Build your own 5 1/4 Drive Connector By Ernest Viveiros
Amiga Basic Tips by Rich Wirch
Scrimper Part One by Perry Kivolowitz A C program to print your Amiga screen
Microsoft CD ROM Conference by Jim O'Keane
Amiga BBS Numbers

Volume 1 Number 5 1986

The HSI to RGB Conversion Tool by Steve Pietrowicz a basic program for color manipulation
AmigaNotes by Rick Rae The first of the Amiga music columns
Sidecar A First Look by John Foust A first "under the hood" look at the IBM compatible hardware
John Foust Talks with R. J. Mical at COMDEX™
How does Sidecar affect the Transformer an interview with Douglas Wyman of Simile
The Commodore Layoffs by John Foust John looks at the "cuts" at Commodore
Scrimper Part Two by Perry Kivolowitz
Marauder reviewed by Rick Wirch
Building Tools by Daniel Kary

Volume 1 Number 6 1986

Temple of Apshai Triology reviewd by Stephen Pietrowicz
The Halley Project: A Mission in our Solar System reviewed by Stephen Pietrowicz
Flow: reviewed by Erv Bobo
Textcraft Plus a First Look by Joe Lowery
How to start your own Amiga User Group by William Simpson
Amiga User Groups
Mailing List by Kelly Kauffman a basic mail list program
Pointer Image Editor by Stephen Pietrowicz
Scrimper: part three by Perry Kivolowitz
Fun With the Amiga Disk Controller by Thom Sterling
Optimize Your AmigaBasic Programs for Speed by Stephen Pietrowicz

Volume 1 Number 7 1986

Aegis Draw: CAD comes to the Amiga by Kelly Adams
Try 3D by Jim Meadows an introduction to 3D graphics
Aegis Images/ Animator: a review by Erv Bobo
Deluxe Video Construction Set reviewed by Joe Lowery
Window requesters in Amiga Basic by Steve Michel
ROT by Colin French a 3D graphics editor

Volume 1 Number 7 1986 (Continued)

"I C What I Think" Ron Peterson with a few C graphic programs
Your Menu Sir! by Bryan D. Catley programming menues in Amiga Basic
IFF Brush to AmigaBasic 'BOB' editor by Michael Swinger Convert IFF Brush Files for use with Amiga Basic
Linking C Programs with Assembler Routines on the Amiga by Gerald Hull

Volume 1 Number 8 1986

The University Amiga By Geoff Gamble Amiga's inroads at Washington State University
MicroEd a look at a one man army for the Amiga
MicroEd, The Lewis and Clark Expedition reviewed by Robert Frizelle
Scribble Version 2.0 a review
Computers in the Classroom by Robert Frizelle
Two for Study by Robert Frizelle a review of Discovery and The Talking Coloring Book
True Basic reviewed by Brad Grier
Using your printer with the Amiga
Marble Madness reviewed by Stephen Pietrowicz
Using Fonts from AmigaBasic by Tim Jones
Screen SaVer by Perry Kivolowitz A monitor protection program in C
Lattice MAKE Utility reviewed by Scott P. Evernden
A Tale of Three EMACS by Steve Poling
.bmap File Reader in Amiga Basic by Tim Jones A look into the .bmap files

Volume 1 Number 9 1986

Instant Music Reviewed by Steve Pietrowicz
Mindwalker Reviewed by Richard Knepper
The Alegra Memory Board Reviewed by Rich Wirch
TxEd Reviewed by Jan and Cliff Kent
Amazing Directory A guide to the sources and resources
Amiga Developers A listing of Suppliers and Developers
Public Domain Catalog A condensed listing of Amicus and Fred Fish PDS Disks
Dos 2 Dos review by Richard Knepper Transfer files from PC/MS-DOS and AmigaBasic
MaxiPlan review by Richard Knepper The Amiga version of Lotus 1-2-3
Gizmoz by reviewed by Peter Wayner A collection of Amiga extras!
The Loan Information Program by Brian Catley basic prog. to for your financial options
Starting Your Own Amiga Related Business
by William Simpson The possible ways to establish your business.
Keep Track of Your Business Usage for Taxes
by James Kummer A program to justify your Amiga to the IRS
The Abesoft Amiga Fortran Compiler
reviewed by Richard A. Reale Use your valuable Fortran programs.
Using Fonts from AmigaBasic, Part Two
by Tim Jones The Amiga Basic program outlined last issue
68000 Macros on the Amiga by Gerald Hull Advance your program's ability.
TDI Modia-2 Amiga Compiler
by Steve Faiwiszewski Looking at an alternative to C and Forth.

Volume 2 Number 1 1987

What Digi-View Is... Or, What Genlock Should Be!
by John Foust A look at the capabilities and liabilities of Amiga Video products
AmigaBASIC Default Colors
by Bryan Catley A look at the default colors on palettes 4 through 31
AmigaBASIC Titles
by Bryan Catley How a program 'looks' goes a long way
A Public Domain Modula-2 System
by Warren Block A look at its faults and great potential
One Drive Compile
by Douglas Lovell Using Lattice C with only one disk drive
A Megabyte without Megabucks
by Chris Erving How to fully expand your 512K Amiga to full megabyte
Digi-View reviewed by Ed Jakober
Defender of the Crown reviewed by Keith Conforti
Leader Board reviewed by Chuck Raudonis
Roundhill Computer Systems' PANEL reviewed by Ray Lance
Digi-Paint previewed by John Foust
Deluxe Paint II from Electronic Arts previewed by John Foust

To Be Continued.....

Your Resource to the Commodore Amiga™

Plus, don't forget our regular columns:

The Amicus Network **(a "Newsletter" of the Amiga Computer Users)**
AmigaNotes **(a music column)**
ROOMERS **(an insider's look at the Amiga Development Community)**
Forth!
The Amazing C Tutorial

Amazing Computing has been offering the Amiga community the best in technical knowledge and reviews for the Commodore-Amiga™ since our first issue in Febuary 1986.

We were the first magazine to document CLI
We were the first to show Sidecar™ from COMDEX™ in full detail.
We were the first to document a 5 1/4 drive connector
We were the first with a 1 Meg Amiga upgrade hardware project!
We were the first magazine to offer serious programming examples and help.
We were the first magazine to offer Public Domain Software at reasonable prices.
We were the first magazine with the user in mind!

However, Amazing Computing™ will not rest on past achievements. The Commodore-Amiga™ has more surprises for you and we are ready to cover them. We even have a few tricks that will "Amaze" you.

To subscribe to Amazing Computing™, please fill out the form below and send to:

PiM Publications Inc.
P.O.Box 869
Fall River, MA 02722.

Back issues are still available at $4.00 each (Foreign orders, please add $1.00 U.S. per issue for P. & H).

Yes! *Amaze* Me!

Please start my subscription to Amazing Computing™ with the next available issue. I have enclosed $24.00 for 12 issues in the U.S. ($30.00 Canada and Mexico, $35.00 overseas) All funds must be in U.S. Currency drawn on a U.S. Bank.

Please circle your selection
Subscription PDS (as noted) Back Issues

Name _____
Street _____
City _____ **St.** ____ **ZIP** _____
Amount enclosed _____

Mass. Residents, please add 5% sales tax on PDS orders

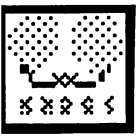
Public Domain Software:
$6.00 each for subscribers (yes, even new ones!)
$7.00 each for non subscribers.

AMICUS: A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14 A15 A16
Fred Fish: FF1 FF2 FF3 FF4 FF5 FF6 FF7 FF8 FF9 FF10
FF11 FF12 FF13 FF14 FF15 FF16 FF17 FF18 FF19 FF20
FF21 FF22 FF23 FF24 FF25 FF26 FF27 FF28 FF29 FF30
FF31 FF32 FF33 FF34 FF35 FF36 FF37 FF38 FF39 FF40
FF41 FF42 FF43 FF44 FF45

BACK ISSUES: $4.00 each (foreign orders add $1.00 each for Postage and Handeling)
VOL.1#1 VOL.1#2 VOL.1#3 VOL.1#4 VOL.1#5 VOL.1#6 VOL.1#7
VOL.1#8 VOL.1#9 VOL.2#1

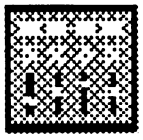
SoundScape...Power Play for the AMIGA.

SoundScape Pro MIDI Studio



The most powerful performance and recording software on any computer. The recording studio-like environment provides complete facilities for routing, recording, editing, transposition and playback of any musical performance. As new modules are introduced, you can "install" them at any time. Music can be performed by the internal sampled sound synthesizer, or with any external MIDI equipment. Record from the QWERTY keyboard or any external MIDI source, including keyboards, guitar and pitch followers. Synchronize with, or provide MIDI clock information, including MIDI Song Pointers. The complete flexibility of the system makes your imagination the only limit to its power.

- Number of notes and tracks determined by available memory
- MIDI patch panel links program modules
- Install new modules at any time
- Up to 16 internal instruments at one time
- Complete sample system with editing, looping, ADSR envelopes, velocity sensitivity, and pitchbend.



- Up to 160 sampled sounds at one time
- Save and load IFF note and sample files
- Quantize to any multiple of MIDI clock beats
- "Match" mode eases learning of a song
- Complete MIDI sequence and song editing
- Route, merge, split, or bounce any track to any other.

SoundScape MIDI Interface

Necessary for any program which supports MIDI to communicate with MIDI equipment.

- Completely compatible with the standard Amiga MIDI interface
- MIDI In, Out, and Thru connectors
- Plugs into the serial port

SoundScape Sound Digitizer

With the SoundScape Sound Digitizer, any sound may be sampled and modified by the Amiga, including voice. IFF File compatibility enables these samples to be used as musical instruments, sound effects, or speech with any IFF compatible music or animation system.

- High quality
- Highest possible fidelity from the Amiga
- Stereo or mono
- Variable sample rates
- Mike and line inputs
- Digitally controlled volume on each channel
- IFF Sample File compatible
- Software included for sampling, editing, and MIDI performance functions

Available From Your AMIGA Dealer.

SoundScape Pro MIDI Studio	**$149.00**
AMIGA MIDI Interface	**$ 49.00**
SoundScape Audio Digitizer	**$ 99.00**

Amiga is a trade mark of Commodore Business Machines

Prices and availability subject to change without notice

mimetics™

corporation **...the professional software source!!**

P.O. Box 60238 Sta. A, Palo Alto, CA 94306 (408) 741-0117

AmigaDOS Version 1.2

"...contains many significant improvements over version 1.1. But, you'd better save those old KICKSTART 1.1 disks."

By Clifford Kent

Compuserve [72437,162]
BIX: ckent
People Link C.KENT

As I write, version 1.2 of AmigaDOS is officially released, but beta test copies have been widely available for some time. I have been using AmigaDOS 1.2 in various forms for several months now. It contains many significant improvements over version 1.1. But, you'd better save those old KICKSTART 1.1 disks.

Some Amiga software won't run with the new DOS, not because of bugs in the new system, but because many things didn't work correctly in the earlier versions of DOS and clever programmers found "work around solutions". Most of these "work arounds" cause problems with the new system, but a few have actually been written into version 1.2 in order to keep major software packages compatible. I expect to see the incompatible software packages upgraded in the near future - AmigaDOS 1.2 is too good to ignore.

This article will describe the changes and new features in the new release. It is available from most Amiga dealers as part of the "Amiga Enhancer Software" package for $14.95.

WORKBENCH

Workbench is now smarter about removing the icons for disks that have been removed from the drive. If a removed disk has no open Workbench windows and no program (including a CLI window) has a file or directory open, the icon will disappear. As an example of how subtle this can get, if you open a CLI window, CD (ChangeDirectory) to df1: then change the disk in df1:, the new icon will appear but the old one will remain - when you do CD df0: the icon for the removed disk will disappear.

It is now possible to COPY the entire Workbench disk to RAM Disk, ASSIGN all the DOS directories (LIBS, DEVS, FONTS, C, L, S, and SYS) to the RAM Disk copy, then use both df0: and df1: as data drives. It will take alot of ram and a long time to boot up, but it will be about the fastest running Amiga you could imagine.

Workbench 1.1's "drag pointer" is gone. When you drag an icon, a copy of the icon follows your pointer. You can even drag several icons at once with "extended selection". Hold down either keyboard <SHIFT> key, click each icon you want to drag, then hold the left mouse button down to drag all of the icons together. This is nice to copy programs to RAM Disk, for example.

There is a new Workbench Drawer, called "Expansion". It is intended to hold device drivers supplied with new Amiga hardware. The CLI command "BindDrivers" adds any drivers found in the Expansion Drawer to the system. If you add the BindDrivers command to your Startup-Sequence, the new hardware will be installed automatically each time you boot up.

Under AmigaDOS 1.1, if you had "fast ram", or other expansion hardware, attached to your Amiga's expansion port, you needed to run an install program before AmigaDOS would recognize the extra space. Under version 1.2 this is totally automatic if your expansion hardware supports the Amiga Auto-Configuration Standard. With 1.2 and auto-configure ram, my Amiga recognizes the 2 meg ram expansion very early in Kickstart and places large segments of the operating system code in "fast ram", thus freeing "chip ram" for use by those video and disk operations that require it.

LoadWB no longer deactivates the current CLI window. This is a small change that will be welcomed by everyone using both Workbench and a CLI window. I suspect it saves me about two dozen mouse clicks a week.

PREFERENCES

The new, expanded Preferences gives more control over the system and cleans up some old problems.

A new window full of serial device options has been added. It allows you to set baud rate (110 thru MIDI), number of read and write bits (7 or 8), number of stop bits (1 or 2), parity (even, odd, or none), handshaking (xon/xoff, rts/cts, or none), and data buffer size (512 thru 16000 bytes).

The old problems with the system clock and "date virus" all seem to be solved. Not only does Preferences set the right time, but the clock in the Preferences window ticks off the minutes as they pass.

The printer window lists only the drivers found on the current Workbench disk. Three new Okidata printers (92, 192, and 292) and the Apple ImageWriter II printer are now supported.

The new "Workbench Interlace On/Off" gadget opens up some new possibilities. The normal Workbench Screen is "medium resolution" (640x200 non-interlace). This allows up to 23 lines of 80 characters (or 21 lines of 60 characters) in a borderless window on the Workbench Screen. By setting Workbench Interlace "On", saving the new Preferences settings, and resetting the machine, you will get a "high resolution" (640x400 interlace) Workbench Screen. An Interlaced Workbench Screen allows up to 48 lines of 80 characters (or 42 lines of 60 characters) in a borderless window.

The interlaced display mode has its price, however. Since the video scan lines are refreshed only 30 times per second (as opposed to 60 times per second in non-interlaced mode) there can be a lot of "flicker" in the displayed image. The amount of flicker will depend on the colors used, brightness and contrast adjustment of the monitor, and the content of the displayed image. If you adjust your monitor for maximum brightness and contrast, then try to display white lines against a black background, the image will almost blink at you. But, if you use dark letters on a light field, use color rather than brightness in the border and cursor color settings, and turn down the monitor's contrast, you may just find the interlaced workbench useful.

To make the interlaced mode really usable, you'll need a long-persistence monitor. This is not necessarily the same as a high-resolution monitor. (As far as I know, none of the "multi-sync" monitors intended for the "EGA" type PC video cards has a long-persistence screen.) I am writing this with a Mitsubishi C-3419LP long-persistence monitor connected to my Amiga. Workbench is set for 60 characters per line with Interlace On. My editor window displays 42 lines of 60 characters, black letters on a light gray page. The characters are a little "squat", but very sharp and easy to read. Interlaced displays CAN look really good.

UTILITIES

AmigaDOS 1.2 also marks the release of Notepad 2.0, a major upgrade from the original Notepad. While still not a real word processor, Notepad 2.0 includes so many improvements that I can only list the highlights here. Notepad now has automatic wordwrap at the right margin. Set the line length by changing the window size. The edit menu now works, including block cut, copy, and paste functions that use the Amiga Clipboard. Also on the edit menu are string search and replace, both forward and backward. Text scroll gadgets have been added to the Notepad Window's right margin, and the handling of fonts has been improved.

Two problems with the Clock Utility have been corrected. The clock window is now moved to the front when the alarm goes off. And the clock size gadget has been fixed - resizing the clock window won't crash the system any more.

STRING GADGETS

The improvements in the operation of the system's text input/editing routines (string gadgets) make many programs easier to use. First, it is now possible for a program to automatically "select" a string gadget. In most programs, you don't have to use the mouse to click in the box before starting to type. Second, when you click in a string gadget, the cursor is placed at the click position. Third, it is possible to make a menu selection while a string gadget is active.

DISK OPERATIONS

Most floppy disk operations are faster under 1.2. The disk heads are stepped faster from track to track - you can hear the difference. Also the disk's physical layout has been changed to minimize the disk's "seek time". The result is faster directory searches. Workbench windows fill with icons faster. Programs and data files load faster. See Figure 1 for instruction in creating faster disks for 1.2.

The RAM disk (RAM:) has been improved. It is generally more reliable and is faster. You can have a Workbench icon for the "RAM Disk" if you add any reference to the RAM: device to your Startup-Sequence file. (Try the line "Dir RAM:".) Once the RAM Disk icon is created, there is no way to get rid of it, although you can delete all the files from it and reclaim nearly all the memory.

RAM Disk is not the only way to improve disk speed. New CLI commands can be a big help. (See Figure 2 for a list of new commands and their syntax.)

The AddBuffers command can be used to give more memory to a disk's driver task. You might start by giving 25k of ram work space to each drive during your Startup-Sequence. (If you have expansion fast ram, you could try 50k or 80k of work space for the drive with your Workbench disk.) You

should find that the disks run less and directory searches and program loads are faster. The disk drivers will use this extra ram to save copies of the disk directory and the files you're using. The disk will run when you ask for something not in the "disk cache", or when you write to the disk. I prefer this disk cache system over a pure ram disk. First, because when I save a data file it is written to the floppy disk almost immediately, making it safe from system crashes and power failures. Second, because I don't have to plan ahead by copying all the things I'll want to the RAM Disk before I start.

The size of the disk cache you can use on the Amiga is limited, however, because the memory used for the cache is "chip memory" - if you add too many buffers, you won't have enough memory left for the video display, which must also use chip memory.

When AmigaDOS 1.1 searched for a file name that did not include drive and directory information, it looked first in the "current" directory (as set with the CD command), then it looked in the "sys:c" or "c:" directory. AmigaDOS 1.2 boots up exactly the same, but the new Path command allows you to customize the AmigaDOS directory search list. For example, if you don't have enough ram to hold all of your CLI utility programs (normally in the c: directory), you can now use the RAM Disk for just those that you use most. First, issue the CLI command "Path ram:", then Copy the programs you use often (like CD, Dir, List, Delete, and Rename) to RAM:. AmigaDOS 1.2 will check the RAM: directory before it checks the c: directory, so, it will load the RAM: copy of the program instead of the floppy disk copy. The result can be very quick response, yet not use up all your ram.

A very welcome addition is the new DiskDoctor CLI command. It can often save nearly everything on a bad disk. DiskDoctor reads through the bad disk, locates the bad area(s), then reconstructs as much as possible. When DiskDoctor has finished with a disk, you simply copy the saved files to another disk. I really like this one - twice so far.

The DiskCopy program has been improved and moved to the System Drawer. If called from CLI it gives the copy the same name as the original. It isn't confused by having more than one disk on the Workbench with the same name. I haven't tested this, but the documents say that it will copy hard disks, disk partitions, 5-1/4" disks, and disks with bad blocks.

A new Format program replaces both the initialize and format programs in AmigaDOS 1.1. It can be called by Workbench or with a command line from a CLI. The new version is in the System Drawer. A new option has been added to its CLI command line. To format a disk with no trashcan icon, you must add "noicons" to the command line. The new version supports hard disks, disk partitions, and 5-1/4" floppy disks.

A SetDate program has been added that makes it possible to arbitrarily change the time stamp on a file with a simple CLI command.

5 1/4 disks

The new CLI command Mount allows you to add new devices to the AmigaDOS device list. To add a new device, you edit its description into the file "devs:MountList". Mount makes it possible to use 5-1/4" Transformer disk drives as limited capacity drives under AmigaDOS.

Most 5-1/4" drives do not provide the "disk eject" signal that AmigaDOS uses to detect disk swaps in the 3.5" drives. The DiskChange command tells AmigaDOS to check a drive for a new disk. Its effect is the same as inserting a new disk in a 3.5" drive.

The Amiga Extras disk contains new utility programs that will read from, write to, or format an MS-DOS disk in the 5-1/4" drive, making it easy to move data files between MS-DOS and AmigaDOS.

Multitasking

The CLI command RUN is a handy way to start a program from the CLI as a new task. Unfortunately, under AmigaDOS 1.1 the RUN command started the new program with a lower multi-tasking priority than the CLI it was started from. This effectively made the new program a background task, AmigaDOS would give it less CPU time than tasks with higher priority. In some cases it didn't matter, but in others the new program was slowed noticeably by its lower priority.

The AmigaDOS 1.2 version of RUN makes no changes in the multi-tasking priorities. A program started by Run has the same priority as the CLI window where the command was typed.

Real task priority control has been added to AmigaDOS 1.2 with the CLI command SetTaskPri. This new command changes the multi-tasking priority of the CLI window (and task) from which it's run. This in turn changes the priority of all the programs started from this CLI window. This may be needed to tame some "undisciplined" programs that used too much CPU time. SetTaskPri could also be used to make one CLI window into a "background" window, used only to start programs that should only execute when there is extra CPU time available, printing a large document for example.

Keyboard mapping

In order to sell Amigas in Europe, the keyboard had to be remapped for each destination country. The SetMap program installs new keymaps into the system. The devs/keymaps drawer contains the data files used by SetMap.

Before selecting a keymap under Workbench, single click the SetMap icon then select "icon" from the "workbench" menu. To select the AmigaDOS 1.1 keymap (needed by a few programs), type "KEYMAP=usa0" (without the quotes) into the TOOL TYPES string gadget, then click the SAVE gadget. From then on, double-click the SetMap icon to install the version 1.1 keymap. You may also want to rename that copy of SetMap so you'll know what it does.

To do the same thing from the CLI or your Startup-Sequence, use the command "sys:system/SetMap usa0".

CLI improvements

Many CLI commands have been improved in small, but welcome ways. Here is a short list of the things I've found.

The DIR command does Amiga style pattern matching on file and directory names. (If only the LIST command used the same syntax.) And the listing can be cancelled with Ctrl-C. The ASSIGN and CD commands will now print full volume and directory names.

ED now does file inserts and window sizing correctly, and handles disk full errors better.

STATUS gives a correct list of the tasks started from the current CLI window (using the RUN command) and from any other CLI windows you open from it (with the NEWCLI command). It does not list tasks that were started from Workbench.

Opinion

AmigaDOS 1.2 is much better than its predecessors, so much better that I would even use an early beta test version of 1.2 (with a few bugs), rather than running 1.1. The release version of 1.2 is faster, more complete, and nearly bug free. However, I hope that development of AmigaDOS will continue. The Amiga hardware provides a unique combination of processing speed, high quality graphics, expandability, and low cost. The system software should continue to improve as the Amiga matures.

continued...

New AmigaDOS commands

AddBuffers df<x>: <nn>
Adds <nn> buffers (512 bytes per buffer) to the work space availble for drive <x>. Adding additional buffers can significantly reduce disk access time, but the buffers must be in "chip ram" so large buffers may not work with graphics programs, even though you have lots of empty "fast ram".

BindDrivers
This command is normally part of a startup script. It is used to add device drivers (found in the directory SYS:Expansion) to the system. For end users, this means that if icons for boards they've added are in the Expansion drawer on the Workbench, the boards will be configured automatically when they boot up.

DiskChange <dr>:
Enter this command to inform AmigaDOS that you've changed disks in drive <dr>:. This is necessary for 5-1/4" disk drives that do not detect when a disk has been changed.

DiskDoctor <dr>:
Use DiskDoctor to save files on the corrupt disk in drive <dr>:. When DiskDoctor is done, use the CLI COPY command to move the important files to a freshly formatted disk.

Format DRIVE <drivename>: NAME <diskname> [NOICONS]
You can initialize any disk or disk partition with this command. Unless you include the NOICONS option, a Trashcan will be added to the disk you initialize. You can stop the command before it is finished by entering CTRL-C or, if you're formatting a floppy disk, by removing the disk from its drive.

The format program is in the directory SYS:system. to use the command, you must specify the full path when you type the command line or add SYS:system to the list of search paths; slowing all program loads slightly.

Mount <dr>:
You use this command to make available a new device, such a 5-1/4" disk drive. For the command to work, there must be an entry for the device in the file devs:Mountlist. This file includes a sample entry you can adapt for different devices.

NewCLI [FROM <filename>]
The new FROM option for the NewCLI command will execute a script file as soon as the new CLI task is created.

Path [SHOW]
Path <directoryname> [, <directoryname> ...]
Path ADD <directoryname> [, <directoryname> ...]
Path RESET [<directoryname> [, <directoryname> ...]]
The path command lets you list or change a CLI window's "search path" - the directories that AmigaDOS searches when looking for a program to execute. By default, AmigaDOS searches the current working directory, then the c: directory for a program. Each CLI task has its own search path.

SetMap <keymap>
Use this command to set the keyboard mapping to a keymap file found in DEVS:keymaps/. The key map in use affects all keyboard operations in all windows. The usa0 keymap file is the same as Workbench 1.1. The usa2 keymap is a Dvorak keyboard. Additional keymaps are provided for ten other countries.

SetTaskPri <priority>
This command changes the multi-tasking priority of the CLI window from which it's run and in turn, tasks started from this CLI inherit its priority. To avoid disrupting important system tasks, enter values from -5 to +5.

SetDate <file> <date> [<time>]
To change the time stamp stored in the directory entry for a file.

continued...

How To Make A Faster Disk

In order to take advantage of the new disk layout used by AmigaDOS 1.2, you will have to make new copies of your 1.1 disks. Diskcopy and other utilities won't do the trick. To get a faster version of an old disk named "Pictures":

1. Kickstart with version 1.2.

2. Be sure to use Workbench 1.2 next.

3. Open a CLI window.

4. Place a blank disk in your outboard drive, df1:.

5. Type the command line:

```
system/format drive df1:name empty noicons
```

Press <RETURN> when prompted for it, then wait for the CLI prompt to return.

6. Type the following five commands in turn:

```
install df1: <CR>
addbuffers df0: 200 <CR>
addbuffers df1: 200 <CR>
copy c:copy to ram: <CR>
cd ram: <CR>
```

7. Remove the Workbench 1.2. disk from df0: and replace it with the disk to be copied, "Pictures".

8. Wait for the df0: drive to stop.

9. Type the command line:

```
copy df0: to df1: all <CR>
```

10. Go for coffee until the CLI prompt returns.

11. Remove "Pictures" from df0: and replace it with Workbench 1.2.

12. Type the command lines:

```
cd df0: <CR>
relabel df1: Pictures <CR>
```

13. Label your new "Pictures" disk and put away the old copy as a backup.

•AC•

AmigaNotes

THE AMAZING MIDI INTERFACE

"...by rolling your own, you can delete bells and whistles you don't want, and add the features you need."

by Richard Rae
CIS [76703,4253]

Last month we took a quick look at the commercially available MIDI interfaces for the Amiga. This month I am delighted to present an alternative: the ***Amazing MIDI Interface.***

Why on Earth would you want to build your own MIDI interface? Well, by rolling your own, you can delete bells and whistles you don't want, and add the features you need. Also, there are a lot of hardware hackers who prefer building and troubleshooting their own hardware to buying prebuilt equipment. And it occurs to me that a lot of programmers might be interested in learning hardware; a simple device like this MIDI interface is an excellent first project. Finally, depending on how valuable you consider your time and how good a scrounger you are, you might even save some money.

The Amazing MIDI Interface, as described here, is fairly flexible. It was designed to minimize the amount of cable swapping needing in a typical small to medium scale synthesizer system. The RS232 input is passed through to a second connector, and a front panel switch selects either MIDI or RS232 operation. This allows you to leave your modem or serial printer connected at all times.

There are two MIDI IN ports, also switch selected. This allows you to connect two master keyboards to your setup and switch between them. A THRU port is provided, which allows you to drive a slave synthesizer with your master keyboard without swapping cables. Two MIDI OUT ports are convenient for those people with no THRU ports on their synthesizers, and help delay the purchase of a THRU box for owners of larger systems.

Finally, a feature I've wanted for some time, but haven't seen on any commercial MIDI interface: the ability to convert the OUT ports to THRU ports. This enables you to play your slave synthesizers directly from a master keyboard, then have the Amiga play a sequence simply by flipping one switch.

The Amazing MIDI Interface was also designed to be simple and easy to build: the entire circuit consists of two integrated circuits and a handful of passive components. Experienced hardware types can probably assemble this circuit in an evening or two.

So much for the sales pitch; let's take a quick look at the schematic. (If you'd like to skip the technish and get right to building the interface, you should bypass the next section.)

THEORY OF OPERATION

The interface receives its power from the Amiga's serial port, and uses both the plus five volt and minus five volt supplies. These come into the interface on pins 21 and 14 respectively; pin 7 is, of course, ground. Note the symbology used on the schematic: an open arrow down is ground, a filled arrow down is minus five volts, and a filled arrow up is plus five volts.

With the exception of pins 2 and 3, ALL the pins on the two DB25 connectors are wired in parallel on the prototype. If you know what device you will be driving with the extension port, you could conceivably wire only the pins it needs. Bussing all the pins together, however, allows you to plug in ANY device you could plug directly into your Amiga, and is strongly recommended.

continued...

Pins 2 and 3 of the input connector are special cases, and go to the common inputs of the DPDT MODE switch. In the RS232 position, these pins are connected to their twins on the output connector, allowing the daisy-chained peripheral device to operate. With the switch in the MIDI position, the fun begins!

Data entering pin 2 from the computer is swinging between RS232 levels, and is applied to two sections of the LM339, which is a quad voltage comparator. All the comparator sections are configured with roughly 10% hysteresis, which means there is approximately a one volt deadband around the threshold voltage. Since the output comparators are referenced to ground, each output will go low when the input signal exceeds about one half volt, and will revert states when the signal drops below roughly one half volt negative; this provides an extra measure of noise immunity.

Each of these comparators drives pin 5 of its associated MIDI OUT jack. This line loops through the receiver in the attached MIDI device and returns to pin 4, which is connected to plus five volts via a 1K ohm resistor. When the comparator output is low a 10 volt drop is developed across this resistor, resulting in 10 milliamps of current flow (minus the protective resistance in the receiver, which is usually only a few hundred ohms). This results in 100% overdrive relative to the MIDI specification, ensuring a good solid signal even with questionable interconnects.

Notice that the THRU circuit is identical to the OUT circuitry, except that it is driven by the RS232 output rather than the input. Thus incoming MIDI data, which has been converted to RS232 for the Amiga, is converted BACK to MIDI and echoed to the THRU port.

Note also that the drive for the output comparators is delivered via the OUTPUT switch. With this switch in the OUT position, the signal comes from the RS232 input as I have just described. With this switch in the THRU position, the signal comes instead from the RS232 output, placing the output sections in parallel with the THRU circuitry and yielding three THRU ports.

The flip side of the interface is just as simple. The desired MIDI input is selected by the DPDT INPUT switch, and then applied to the input of the 6N138 optoisolator through a protective resistor. This resistor, in conjunction with the reverse-biased diode, helps prevent destructive currents in the optoisolator in the event of incorrect external connections (i.e., a reverse-wired MIDI cable).

When a current flows through the input diode of the optoisolator, it excites the output stage, pulling pin 6 low. When the current is interrupted, the 10K output resistor returns pin 6 to plus five volts.

Since the output of the optoisolator only swings between five volts and ground, the input comparator is referenced to 2.5 volts via the two 22K resistors. Thus its output goes low with an input above roughly three volts, and reverts state with an input below about two volts.

The LM339 comparator is an open collector device, so a 2.2K ohm pullup resistor is supplied to provide the high RS232 level. The resulting signal drives the THRU comparator as described earlier, and is also routed through the MODE switch to pin 3 of the DB25 input connector. The final result is an RS232 signal which swings between plus and minus five volts, which is quite adequate for the Amiga.

Notice that the ground pins of the MIDI IN ports are floating. This is in accordance with the MIDI specification, and makes it impossible to develop a ground loop through the MIDI cables.

And that, in a rather technical nutshell, is all there is to the interface.

BUILDING IT

If the schematic looks like so much Greek to you, or you just don't know where to start, find a friend willing to help you and dive in! This is a very good first project, even for people who don't know which end of a soldering iron to hold. Don't be ashamed to ask... remember that we all began where you are now.

The circuit is not critical in any way, although you can damage some components (or your computer) with incorrect connections. Double check you wiring before applying power the first time.

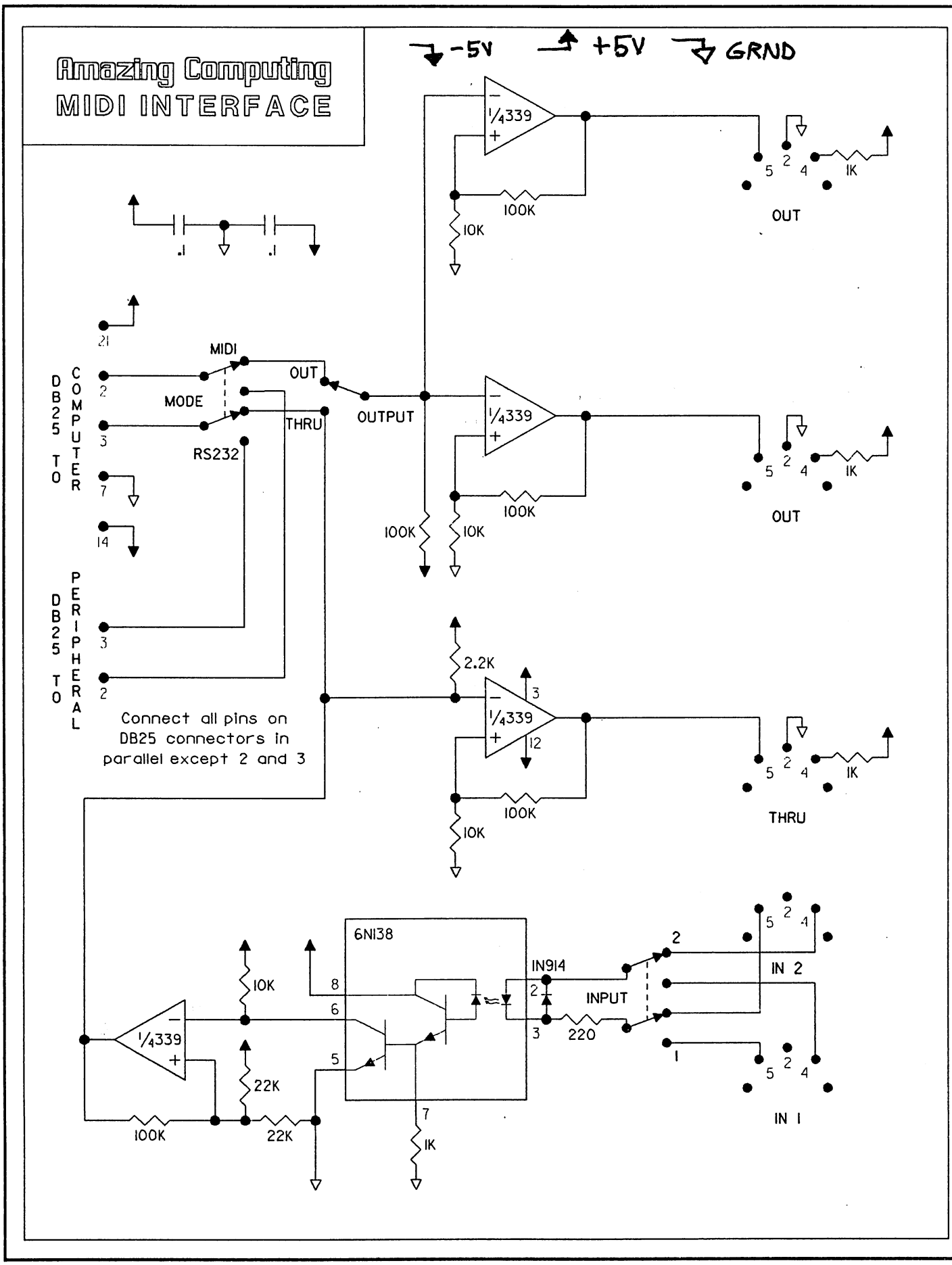
The only sensitive component is the 6N138, which is susceptible to damage from static electricity. Don't fondle your optoisolator; keep it in the factory carrier until you are ready to plug it into its socket.

The only part you can expect having problems finding is - you guessed it - the 6N138. (I expect it's one of Murphy's Laws that the most expensive, least available part in a circuit will always be the easiest one to damage.) This device is made by HP and TRW and is easily available through large distributors, but you won't be able to walk into your local Radio Shack and pull one off the pegboard. Try Hallmark Electronics, Hamilton- Avnet, or Schweber Electronics; these are all HP distributors. You should be able to find the 6N138 for roughly $2 to $3. If anyone finds a readily available source, I'm sure we'd all appreciate hearing about it. All the other parts, including the MIDI connectors, ARE available just about anywhere that carries a reasonable selection of electronic components.

This circuit is simple enough for wirewrapping to be a quick, viable construction method. Discrete components can be soldered to headers and plugged into wirewrap sockets; this was the method used on the prototype. If you have easy access to the needed facilities you might want to get fancy and make a PC board, but this is overkill unless you are planning to make several units.

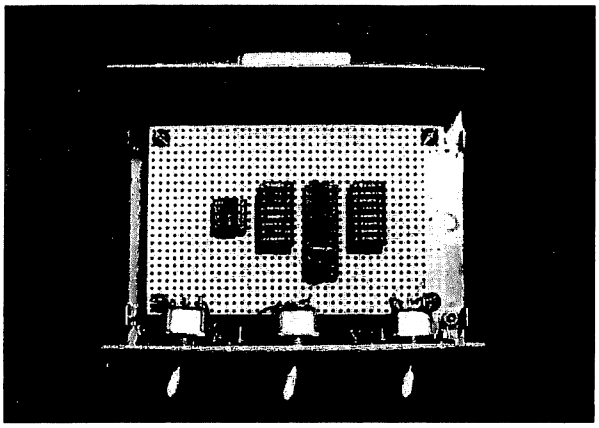
The board may be housed in just about any available enclosure, since the box is not used electrically. Using a metal enclosure might not be a bad idea, though, if you are

continued...


Amazing Computing
MIDI INTERFACE
-5V
+5V
GRND
1/4339
100K
10K
IK
OUT
.1
.1
21
MIDI
OUT
MODE
THRU
OUTPUT
RS232
DB25 TO COMPUTER
DB25 TO PERIPHERAL
2.2K
Connect all pins on DB25 connectors in parallel except 2 and 3
6N138
IN914
INPUT
220
IN 2
IN 1
22K

trying to keep your RF emissions down. The mounting of the electronics and external parts is entirely up to you; the prototype illustrates but one approach.

Figure 1
Interior view of the Amazing Computing MIDI Interface.



As you can see, the circuit is relatively simple. Note that the prototype was wirewrapped, so the discrete components are mounted in IC sockets on component carriers.

ONE FROM COLUMN A AND TWO FROM COLUMN B

One of the advantages of homebrew, like programming, is that you can make your projects do exactly what YOU want them to. With that in mind, here are a few suggestions you might want to consider.

The prototype has two DB25s -- one male and one female -- mounted on the back panel. The peripheral device will only plug into the connector which matches the Amiga's serial port, making it nearly impossible to plug into the wrong one. A 25 wire male to female RS232 ribbon cable connects the interface to the Amiga. This has a hidden advantage: if you ever need a serial extension cable you can, in a pinch, steal your interface cable and use that.

If you are trying to keep costs to a minimum, however, you could dispense with this and simply hardwire cables to the box. The DB25 connectors on the ends of the cables would then plug directly into your Amiga and peripheral. This also eliminates the need to cut those fancy rectangular holes for mounting the connectors.

Notice in the interior photo that the DB25s are tied together with individual wires. Unless you are a masochist, I advise strongly against this... 50 additional solder connections can take a lot of the fun out of building this device. Use the clamp-on ribbon type connectors and save yourself lots of needless effort.

You can reduce the complexity of the circuit to keep costs down. If you know you'll never need a THRU output, for example, you can eliminate three resistors, a MIDI connector, and a switch.

Going in the other direction is possible too: you can build the basic circuit up to do whatever you need. Just keep two guidelines in mind: don't try to draw an excessive amount of current from the Amiga's supplies, and drive only one MIDI line with each comparator output.

One other useful modification might be to have three output switches, one for each output jack. This allows any combination of THRU and OUT ports as needed. We'll take a look at why this might be useful right now...

TYPICAL CONFIGURATIONS

So, you've built the interface, no problems along the way, it works perfectly. Now what? Well, you certainly know you can hook the MIDI OUT to the MIDI IN of your synthesizer and play Music Studio tunes, or connect MIDI IN to MIDI OUT on your keyboard and record in realtime with SoundScape ProMIDI, but what beyond that? Let's take a look at a couple of hookups which might make life easier.

Figure 2A (*see next page*) shows a fairly versatile setup with two synthesizers. With the OUTPUT switch in the OUT position, you can play a tune on the master synthesizer and record it with the Amiga. You can also play back sequences from the Amiga and have both synthesizers play their parts.

Flipping the OUTPUT mode switch to THRU allows you to play the slave directly from the master keyboard. Note, though, that the OUT of the master is connected back to the IN of the master, via the output which was just switched to a THRU. Since playing the master keyboard already sounds the notes, this loop causes TWO voices to be assigned to each keypress. Depending on the synthesizer and patch selected, this can create some very nice doubling effects. It also cuts the number of simultaneous voices in half, so this is generally not useful.

This problem can be avoided in several ways. You can turn the master keyboard's local control off, which effectively disconnects the keyboard from the sound generators so they only respond to the data coming back on the MIDI line. You can set the synthesizer to transmit and receive on different channels, so that it ignores its own data. Or, with separate switches for each output, you can switch only the slave synthesizer's port to THRU mode.

Figure 2B is identical, with the addition of one cable. This comes in handy when you have compatible synthesizers and would like to swap patches between units, or program one from the other. It is especially useful with "black box" expanders, such as the Yamaha TF1 or TX7, which can only be programmed with an external synthesizer.

With the input switch set to IN 1, this configuration allows you to do everything described previously, with the attendant limitations. By flipping the input switch to IN 2, however, the master and slave exchange roles, and you can easily dump patches out of the (original) slave and into the (original) master.

continued...

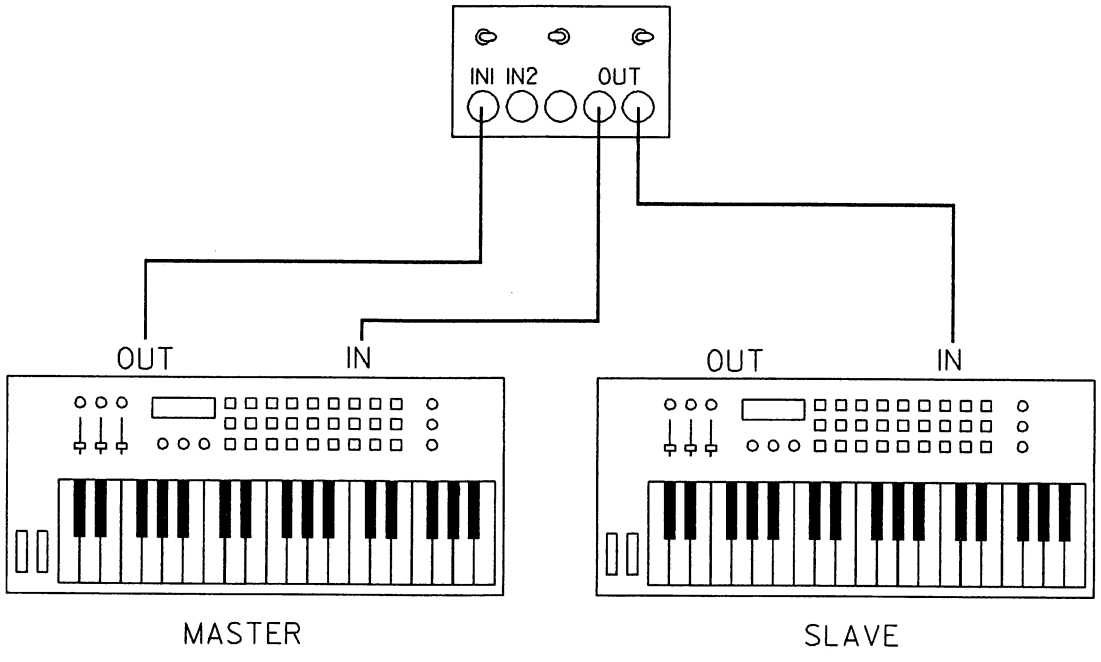

IN1 IN2 OUT
OUT IN
OUT IN
MASTER
SLAVE

FIG 2A

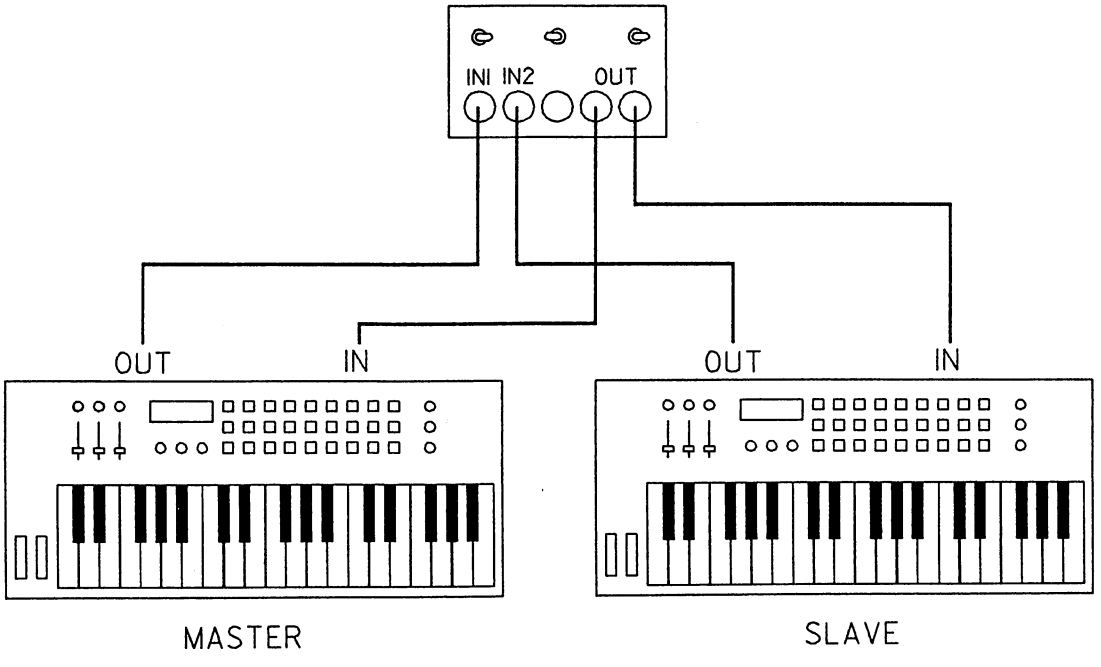

IN1 IN2 OUT
OUT IN
OUT IN
MASTER
SLAVE

FIG 2B

One warning... do NOT flip any of the switches while MIDI data is going through the interface. This is absolutely, completely harmless and won't hurt a thing in hardware or software. But if you flip the wrong switch after a Note On and before the associated Note Off, it's stuck note time!

Of course, now that I've told you not to do something, I have to tell you why you might WANT to; there are some creative uses for this phenomenon. For example, using the interface as a THRU box, you can apply infinite sustain to a chord by playing the notes and flipping the input switch before releasing the chord. Or, you can transpose a slave keyboard by applying a pitchbend at the master, changing inputs, releasing the pitchbend, and switching back to the original input. The slave synthesizer will continue to play the interval you dialed in as long as you don't touch the master keyboard's pitchbend wheel. Nudging it ever so slightly will immediately snap the slave back to it's original key.

That's going to do it for this month, but before I sign off I want to thank Ken Rummage for his assistance on this project; his knowledge and effort were essential in determining the optimum optoisolator configuration.

As always, I'm hoping to hear from you about what you want to see in this column, as well as what you're doing in the audio field. Drop me some mail and let's chat.

Nybbles,
Rick

PARTS LIST

RS 276-154A

Resistors *(1/4 watt 5% or better)*
1 220 Ω
4 1K Ω
1 2.2K Ω
4 10K Ω
2 22K Ω
5 100K Ω

Capacitors
2 .1 µf 12V disc

Diodes
1 1N914 or 1N4148

Integrated Circuits
1 6N138 Optoisolator
1 LM339 Quad Comparator

Hardware
1 SPDT Switch
2 DPDT Switches
1 DB25P RS232 Connector
1 DB25S RS232 Connector
5 MIDI Jacks - 5 Pin 180 Degree DIN Sockets
1 RS232 Cable - 25 Line Straight Through Extension

Miscellaneous
Depending on your construction method, you may need IC sockets, component headers, perf or pc board, a chassis box, standoffs, and other assorted goodies.

•AC•

AmigaDOS Operating System Calls and Disk File Management

"What's actually happening when you click on that icon?"

by Dave Haynie

Usenet: {ihnp4,allegra,caip}!cbmvax!daveh
CIS: [76703,2047]
QuantumLink: Hazy

All Amiga users are somewhat familiar with files on the Amiga, especially disk files. In normal use, several layers of software isolate the user from the AmigaDOS file structure and the things that are actually happening in order to read and write to the disk or send something to the printer.

This is ultimately related to the physical hardware of the Amiga, but were it very dependent on the hardware it would be a poor disk operating system, or DOS. The function of any DOS is to isolate the computer's user from the physical properties of the filing system.

A good DOS will keep the user from the information he/she doesn't need to be concerned with, and manage that information automatically in a standard way. At the same time, a good DOS should allow the user to strip off an arbitrary amount of this isolation, to find a level of usage that best suits the task at hand.

AmigaDOS is a name that crops up in several places in reference to the Amiga system software. Anyone using a CLI encounters "AmigaDOS" as a system of utility programs that make up an interactive disk operating system manager. These programs aren't really all of AmigaDOS, however.

Each of these utilities is a rather simple program that glues the features of the CLI or Workbench to one or more of the functions provided by the underlying AmigaDOS software subsystem. This AmigaDOS subsystem, a set of functions roughly analogous to the set of CLI commands, is used by nearly every program that runs on the Amiga to manage files (and more) in a consistant way.

Although you may have never used anything on your Amiga other than the Workbench interface, you've come in contact with the underlying AmigaDOS routines, though you may not have realized it.

WHAT A USER KNOWS ABOUT FILES

The Amiga's floppy disk, as the only permanent storage device on the basic Amiga, is important for any operation; ordinarily something from two separate disks (KickStart and some kind of WorkBench disk) must be loaded in order to do anything on the Amiga.

Even if the application runs under Workbench, interaction with the Amiga floppy drive to load Tools and Projects from disk is required. All the user actually does is double click the mouse on an icon to load it; no concern about disk files is required; the Workbench program turns all of the clicking operations into commands for AmigaDOS, the underlying filing system.

CLI deals with commands and files instead of Tools, Projects, and Icons, but the effect is the same; there's little concern for what's actually in a file, other than the large abstractions of "program", "spreadsheet", "letter", etc.

All of these are files, and they aren't necessarily even on a disk; they could, for instance, be in the RAM: disk emulator, a CON: window on the Amiga screen, or PRT:, the system printer. (Amiga software release 1.2 will support RAM: from Workbench; currently RAM: is accessable only at the CLI level),

At the user's level, a disk file is a member of the abstract class of items called "files", and all that's really important is that a disk file is something that preserves data from session to session.

What's actually happening when you click on that icon, however, has alot to do with the upper levels of AmigaDOS. One of AmigaDOS's jobs is to load executable code; some form of 68000 machine language that can run as a program.

When you click on a Tool icon, the Workbench system recognizes those clicks, and translates them into the proper AmigaDOS function call. Similarly, the CLI recognizes the commands you type as requests to load programs, and translates these requests into the same AmigaDOS function call. AmigaDOS is normally the only agency in the entire system that will load executable machine language.

Every executable machine language program will be stored on disk, in an AmigaDOS managed object called a file, as mentioned above. Workbench users see these occasionally, CLI users more often. All files are managed by AmigaDOS, and there are only a few kinds. There are, for example, two distinct kinds of disk files, directories and data files. A directory, called a drawer in Workbench, is essentially a unit of organization, it allows other directory files and/or data files to be grouped together.

A data file is the highest level unit of storage under AmigaDOS. Although it often appears otherwise, there is only one kind of disk based data file. What differentiates a program file from a text file is the internal structure of that file. This is a powerful mechanism, because it allows a single set of AmigaDOS functions to operate on all files, no matter what they are designed to represent. It also places the emphasis of file typing on individual applications instead of AmigaDOS itself.

An example of an internally type file is the executable file. It looks the same as any other file, and externally it is. Internally, it contains information and structure that has special significance to AmigaDOS only. Similarly, icons are another file with special internal structure.

The WorkBench program is designed to recognize this special structure and extract the icon image, the deault tool, etc. Many programs running on the Amiga require some kind of input file, usually in an internal format that's special to that program, though on the outside still just another AmigaDOS file.

This input file could be a simple ASCII text file, like a C program or the text file that I'm currently typing, or it could be an IFF graphic format file for a painting program. But on the outside its just a file, nothing special. And this allow AmigaDOS to provide a set of standard tools that a programmer can use, no matter what the internals of his application file look like.

PROGRAMMING FROM A HIGH-LEVEL LANGUAGE

AmigaDOS programs are written with some kind of language, which is either a high-level language, like Modula2, C, LISP, etc., or the native 68000 assembly language. And, as I mentioned previously, many programs do some kind of file oriented input and output.

So, logically, your programming language is bound to provide some mechanism to access the internal elements of files, not just the file as a single entity. The programmer moves one abstraction level closer to what actually constitutes a file.

This level still provides isolation from the physical properties of a file; the programmer names file the same way a CLI user would, and won't usually care whether the file is coming from RAM:, going to PRT:, or on a disk. It is still AmigaDOS providing this abstraction; the language's native functions are built on top of the basic AmigaDOS routines.

The power of this system is that, since AmigaDOS abstracts what's physically happening, the programmer doesn't have to be concerned with making his program work with the printer, the ram disk, the floppy, the hard disk, and the new thing that's added a year later. The programmer writes an program that works with a file, AmigaDOS makes everything old and new a standard file.

What the language provides are functions that let the programmer read and write data that's in some of the abstract data types provided by the language. Just as AmigaDOS can't anticipate the internal structure of various files, it can't easily anticipate the various kinds of data types that different languages provide. This is left completely up to the standard language functions. Some simple file functions at this level are:

Function	C	Modula2	LISP
Open a file	fopen()	OpenInput()	(open)
Write to a file	fprintf()	WriteString()	(print)
Read from a file	fscanf()	ReadString()	(read)
Close an open file	fclose()	CloseInput()	(close)

The C functions all refer to an explicit data object called FILE. Modula2 and Cambridge LISP refer to an input and output "stream", which starts out as the screen and keyboard (CON:), and can be "redirected" to go to or come from another kind of file, like a disk file.

Each language shown here has functions that interact with files in various different ways. C language can read and write all simple data types with the functions shown, but it also has functions that handle data types on an individual basis.

Thus, in C I could write "fprintf(file, "%s", string)", using a general function to write out a string, or I could write "fputs(string, file)", using a specific string function to get the same result. Modula2 has individual read and write functions for each simple type it supports, but no general function like C does.

LISP can read only single characters or its higher level structured constructs ATOMs and LISTs. Each of languages also possess some facilities to convert their natural input/output data types to and from other internal data types.

What all of these functions have in common is that they're abstracted at the level of data and of file. As in the CLI, AmigaDOS provides the file abstraction. The language implementation isolates the specifics of the AmigaDOS operating system from the user, making the code written portable between different computers supporting the same language.

Every language can use its standard functions and data type very easily on the Amiga, and tomorrow the same program can be transferred and compiled on a completely different computer supporting that same language. The tradeoff for this standardization can often be twofold. The extra software overhead of the language specific routines can be a burden, especially in the case where a similar routine exists as a DOS function.

The other problem is that a language cannot usually anticipate the special features of a computer, so using nothing other than the standard functions of a language can seriously underutilize the Amiga. Fortunately, in the case of the DOS as well as most other things on the Amiga, the Amiga specific routines can be directly accessed.

SIMPLE FILE ORIENTED DOS CALLS

Stripping the high-level language software from the DOS interface, the programmer has direct use of AmigaDOS system calls for file access. All of the languages that run on the Amiga will ultimately access these operating system functions, though a casual programmer will probably never realize this. The layer of abstraction that you lose is that of the language's data types; all DOS level input and output is simple in terms of blocks of untyped data bytes.

Figure 1

Standard C Call	AmigaDOS Call

`FILE *fopen(name, mode)`
`  char *name;`
`  char *mode;`
	`struct FileHandle *Open(name, mode)`
	`  char *name;`
	`  long mode;`

`int fprintf(f, format [,adat])`
`  FILE *f;`
`  char *format;`
`  [ATYPE adat;]`
	`long Write(f, buffer, length)`
	`struct FileHandle *f;`
	`  char *buffer;`
	`  long length;`

`int fscanf(f, format, [,adat])`
`  FILE *f;`
`  char *format;`
`  [ATYPE *adat;]`
	`long Read(f, buffer, length)`
	`struct FileHandle *f;`
	`  char *buffer;`
	`  long length;`

`int fseek(f, offset, mode)`
`  FILE *f;`
`  long offset;`
`  int mode;`
	`long Seek(f, offset, mode)`
	`struct FileHandle *f;`
	`  long offset, mode;`

`int fclose(f)` `  FILE *f;`	`long Close(f)` `struct FileHandle *f;`

Any C program using the DOS calls should be sure to contain the line '#include <libraries/dosextens.h>', which should in turn include anything required for access to the libraries. Other languages will have equivalent methods for including the proper data information.

All of the DOS functions are found in an Amiga Resident ROM Library, however, it probably won't be necessary to explicitly open this library from a high-level language - it certainly isn't in Lattice C. The reason for this is that any language on the Amiga must use DOS calls for its built-in I/O functions, so the language itself will in many cases open the DOS library for you.

These functions are like the "natural language interface" for Amiga Assembly language, since there are no standard I/O functions in Assembly as there are in high level languages. Most modern operating systems support file I/O functions; the basic functions that AmigaDOS supplies are Open(), to open a file, Write(), to write to a file, Read(), to read from a file, Close(), to close a file, and Seek(), for random file access.

The natural interfaces to any of the higher level languages will be written in terms of these primitives, which in reality are reasonably high level themselves.

Although the natural interface calls provide language portability and data abstraction, the AmigaDOS calls provide a minimum level of portability between languages running on the Amiga. It would be conceivable to use these same calls directly from every language on the Amiga, though in practice this is not done.

First of all, some languages aren't designed to easily hook into such DOS calls. And for those that do, the loss of data abstraction for input and output would result in lots of extra work in many applications. It is the place of the high level language to convert its data type into byte oriented data that can be read or written with these OS calls, and its the job of the programmer to decide which level his program should operate on.

A comparison of the different levels using C code follows shortly. The C constructs are probably very familiar to most C users, and require the <stdio.h> file to be included. The the DOS structures are defined in the include files dosextens.h and dos.h and will be explained in detail later.

For non-C programmers, the "char" type is an 8 bit character, the "int" type is an integer with an implementation dependent length, the "long" type is a 32 bit integer, the "*" character is read as "pointer-to", and the "struct" keyword indicates a user-defined structured varible (analogous to RECORD TYPEs in Pascal or Modula2) many of which are required by the Amiga system calls.

The above examples differ in a few ways. First off, all C operations deal with a C specific abstraction of "file pointer", a pointer to a thing called FILE. The DOS has its own specific abstraction of "file pointer", a pointer to an object known as a FileHandle. The FILE object in this case is a superset of the DOS FileHandle, containing things important to C as well as AmigaDOS.

On another machine FILE could be a thing as simple as a pointer to an integer containing a device number; any transported programs with C functions would still work, while

the DOS calls would have to be rewritten for the new machine's DOS. The other main difference, as mentioned above, is the level of data abstraction. The C functions require a string of characters that specify each of the elements to read or write, followed by a list of these elements; any number may be specified, and their types may be mixed.

The DOS functions operate only on individual byte buffers of any length, and even this length must be explicitly specified. The DOS supports no data abstraction; anything special about this data is known only to the programmer and his application program. DOS files can be opened with mode MODE_OLDFILE for currently existing files or MODE_NEWFILE for new files.

As shown above, the basic "file" unit in AmigaDOS is the FileHandle. Its internal structure (which is defined to C users in the aforementioned include files) is a series of 11 long words, which appear sequentially (*see figure 2*).

Figure 2

struct Message * fh_Link	Exec message, unused in 1.1
struct MsgPort * fh_Port	Communication ports to handlers, based on the type of file opened.
struct MsgPort * fh_Type	
LONG fh_Buf	Pointer to data buffer.
LONG fh_Pos	Position in buffer.
LONG fh_End	Pointer to end of buffer.
LONG fh_Funcs	Pointers to functions to call for buffer empty, buffer full, and buffer closed.
LONG fh_Func2	
LONG fh_Func3	
LONG fh_Args	Arguments to pass, based on buffer type.
LONG fh_Arg2	

This structure is returned as a BCPL pointer by the Open() function, and its accepted as a file pointer by all the file oriented DOS calls. A BCPL pointer is a longword aligned memory pointer that's been shifted two places to the right. This comes from the fact that much of the AmigaDOS subsystem was written in BCPL, a language that required a longword oriented computer to properly run.

Thus, before examining the internal structure of this structure, it must be shifted back left two places. The resulting pointer is a normal longword aligned pointer that can be examined as any other pointer can. There's very little need to examine the internal structure of a FileHandle during normal programming. The FileHandle's internal structure matters only if you are doing advanced DOS programming, like asynchronous I/O to a device (which is complex enough be beyond the scope of this article).

AmigaDOS supports a few more calls based on simple files. Some of these are available in some form or another in high level languages, but others are too closely related to the operating system to normally show up in a high level language. These functions, represented in C, are:

```
struct FileHandle *Input();

struct FileHandle *Output();

long IsInteractive(f)
   struct FileHandle *f;

long WaitForChar(f, time)
    struct FileHandle *f;
    long time;

long Seek(f, position, mode);
    struct FileHandle *f;
    long position, mode;

long IoErr();
```

The first two functions return FileHandles to the equivalents of C's "stdin" and "stdout" or the default input and output streams in LISP or Modula 2. These are very often a keyboard and a display window, respectively, but via file redirection they can become any system valid file.

The IsInteractive() function is an aid in examining such a file; it will return -1 if the file is a "virtual terminal" of some kind (like a keyboard), otherwise it will return 0, such as in the case of a disk file.

The WaitForChar() function lets a program "go to sleep" for a specified period of time (in microseconds) while waiting for an input from such an interactive terminal. It returns a -1 if a character is available to be read in with Read(), and a 0 for failure. This wait is friendly to multitasking; it consumes very little processor time waiting.

The Seek() function is similar to C's seek() function; it accepts a FileHandle and a position (offset) number, which is measured in bytes. The function will move the file pointer based on this position and the mode, being OFFSET_BEGINNING (move relative to the file's start), OFFSET_CURRENT (move relative to the current position in the file), or OFFSET_END (relative to the end of the file). The function returns the old position of the file, relative to the file's beginning.

The IoErr() call is applicable to all AmigaDOS functions, not just the file oriented variety, but its can be discussed here anyway. This function is used to determine the class of an

event, usually an error, signaled by a DOS function of any kind. Most functions, Open() for example, have some way of signaling an error; Open() itself returns 0 if unsuccessful. Calling IoErr() after receiving this 0 allows the exact cause of the error to be discovered. The value of IoErr() after a successful DOS call seems to be useless; one should always write code (in C) like:

```
if (Open(file,MODE_OLDFILE) == 0 &&
        IoErr() == ERROR_OF_SOME_KIND) {...}
```

Note that the C "&&" (boolean AND) operator only evaluates its second argument if the first evaluates non-zero. The LISP (AND) function is also this so-called "short-circuit" AND, but languages like Pascal would require a nested IF construct to achieve the same effect. The include files <libraries/dosextens.h> and <libraries/dos.h> define many different error conditions. A few of the more common errors returned are:

```
File:
ERROR_OBJECT_IN_USE          File exclusive to another process.
ERROR_OBJECT_EXISTS          Attempted illegal over-writing.
ERROR_OBJECT_NOT_FOUND       It wasn't there.

Disk:
ERROR_DISK_WRITE_PROTECTED   Check the Write Protect tab.
ERROR_DISK_FULL              No room left on this one.
ERROR_NOT_A_DOS_DISK         Could be Kickstart, blank, or MS-DOS
ERROR_NO_DISK                Empty drive
ERROR_NO_MORE_ENTRIES        Last file in a directory
```

The functions I've described so far have similar counterparts as native functions in most languages. These are pretty straightforward, used very often, and tend to be machine independent, which is why they were incorporated into these languages in the first place. Because of this, they are not significantly more useful than the language supplied equivalents.

The native language supplied utilities are a bit easier to use, but take up more memory; the AmigaDOS functions are smaller and faster. This is a certainty, since the native functions of any Amiga language are going to use the AmigaDOS system calls as their foundation. Whether one or the other is more efficient will depend heavily on the desired application.

There is a group of other AmigaDOS functions, however, that provide utilities that are Amiga specific and is thus beyond the scope of most languages. These functions are the main reason to use direct AmigaDOS calls.

FILE LOCK FUNCTIONS

The FileHandle is useful for many disk file operations, as well as operations on arbitrary types of files. AmigaDOS, however, provides another kind of file access, the FileLock, which is designed specifically to provide special access to disk files.

FileLocks don't require the same overhead that FileHandles do, and they allow files to be shared by different processes (only in read mode; several processess writing the same file would be chaos, and so this is forbidden). These objects also have related functions that provide quite a bit of useful information about disk based files. The data structure of a FileLock consists of 5 longwords, organized and named as in figure 3.

The structure reveals the fact that this is obviously intended for disk files; the fl_Key field, for example, is a pointer to the physical location of the file or directory header, on the actual disk.

The functions that operate on FileLocks as well are oriented for something that exists in some kind of disk structured device. Note that all the functions, like the FileHandle functions, actually deal with structure pointers, not the structures themselves.

A few simple DOS functions create FileLocks. Note that only one function can create a FileHandle, that's the Open() function.

Figure 3

```
-----------------------------------------
|    BPTR                fl_Link        | A link to next lock in the
-----------------------------------------   system chain.
|    LONG        fl_Key                 | Physical block number of disk
-----------------------------------------   file.
|    LONG        fl_Access              | Type of access (shared or
-----------------------------------------   exclusive.
|    (struct) MsgPort *  fl_Task        | Pointer to message port to
-----------------------------------------   owner task.
|    BPTR                fl_Volume      | Pointer to disk volume.
-----------------------------------------
```

There are five lock producing functions available. AmigaDOS uses locks as kind of a general file pointer for many operations, since they require less overhead than FileHandles, they permit multitasking file access, and the functions available that operate on them are designed to handle files as whole objects, not element by element (and because of this, most of the AmigaDOS CLI commands can be written using these functions).

The FileLock producing functions are:

```
struct FileLock *Lock(name, mode)
    char *name;  long mode;

struct FileLock *CreateDir(name)
    char *name;

struct FileLock *DupLock(lock)
    struct FileLock *lock;

struct FileLock *ParentDir(lock)
    struct FileLock *lock;

struct FileLock *CurrentDir(lock)
    struct FileLock *lock;

long UnLock()
    struct FileLock *lock;
```

These functions are much simpler than they look. The Lock() function simply returns a FileLock pointer based on the C-style ASCII string representation of that file. This is the FileLock analog to the FileHandle Open() function. The access modes are ACCESS_READ (shared) or ACCESS_WRITE (exclusive).

The CreateDir() functions returns a FileLock pointer to a directory it creates based on the C-style string passed. The DupLock() function returns a copy of the passed lock for a shared access lock. The ParentDir() function returns a lock to the parent directory of the passed lock.

The CurrentDir() function accepts a lock to a directory and makes that directory the current directory, returning a lock to the previous current directory. Finally, the UnLock() function removes a file lock returned by one of these other fuctions.

There are three functions that operate on FileLocks, and each is designed to deliver information. This information is returned in one of two possible DOS structures, the "InfoData" structure or the "FileInfoBlock" structure, both of which MUST be longword aligned. See Listing One for an example of this.

The InfoData structure is filled by the Info() function, which is passed a valid FileLock pointer and an InfoData pointer, and returns 1 if successful, 0 otherwise. The information is based on the particular disk volume referenced by the lock, much of which is what you see when you issue the CLI "Info" command. An InfoData structure is a series of nine longwords (*see figure 4*).

Figure 4

LONG	id_NumSoftErrors	The number of errors found.
LONG	id_UnitNumber	The physical unit number.
LONG	id_DiskState	Disk status, could be protected.
LONG	id_NumBlocks	Total block size of the disk.
LONG	id_NumBlocksUsed	Number of blocks in use.
LONG	id_BytesPerBlock	The size of each block.
LONG	id_DiskType	DOS, Kickstart, BAD, etc.
BPTR	id_VolumeNode	BCPL pointer to volume node.
LONG	id_InUse	TRUE if its actually in the machine.

This structure can be found via the Info() function for any disk that the Amiga is still remembering, which includes any disks in any disk drive as well as any disk that was open and still has valid locks open on it. The programmer should always qualify the information with the id_InUse field to make sure that the disk is actually present; otherwise the information isn't necessarily TRUE (protection and disk unit number can easily change; if the disk is going between two Amigas for some reason, anything could possibly change when the disk is removed).

The FileInfoBlock (FIB) structure is an analogous structure for disk files. It allows the various elements of disk files to be closely examined.

The functions Examine() and ExNext() both use this structure. Both functions are similar; each expects a valid FileLock pointer and a pointer to a FIB structure and returns a 1 for success and a 0 for failure. Examine() fills an empty FIB from a FileLock; the ExNext() function takes a filled-in FIB and updates it with a FIB for the next directory entry. Both of these functions are used in the example program that follows.

A FIB contains the following information, organized as 7 longwords and 2 character arrays (*see figure 5*) Thus, just about any specific information on a file can be derived from a FileLock, which in turn can be derieved from a file name string. There are functions for setting some of these items as well, and a few more DOS functions that operate on files as a whole.

Figure 5

LONG	fib_DiskKey	Physical header block key.
LONG	fib_DirEntryType	Type of Directory. If < 0, a file, > 0, a directory.
char	fib_FileName[108]	C-style string, only 30 chars currently used.
LONG	fib_Protection	Protection bits, 0-3 are used currently.
LONG	fib_EntryType	
LONG	fib_Size	Number of bytes in file.
LONG	fib_NumBlocks	Number of blocks in file.
struct DateStamp	fib_Date	Date file last changed.
char	fib_Comment[116]	The file comment, as a C style string.

STRING BASED DOS CALLS

The one common factor among these remaining DOS calls is that they all operate on files specified as C-style (ASCII NUL terminated) strings, instead of the more complicates FileLock or FileHandle structures.

The main advantage to specifying a string instead of another structure, like a FileLock, is that in many cases, the operation performed is the only thing you'd want to do with that file. If the function accepts the name, only that one function must be called If instead the function accepted a FileLock, then a Lock() call, followed by the function call, followed by an UnLock() call would be required. These functions are, ideally, something that would be using the referenced file only once.

The functions, as specified in C, are:

```
BOOL Rename(oldname, newname)
    char *oldname, *newname;

BOOL SetComment(name, comment)
    char *name, *comment;

BOOL SetProtection(name, mask)
    char *name;
    long mask;

BOOL DeleteFile(name)
    char *name;
```

This first function allows you to set the value of the "fib_FileName", duplicating the function of the CLI "Rename" command. This Rename() accepts C-strings and sets the name of the first argument (the actual file) to the second string. If the first doesn't exist, the second does exist, or the two names point to different devices, the function will fail, returning FALSE. Otherwise it will return TRUE. These names are currently supported as 30 characters long.

The next function, SetComment(), sets the "fib_Comment" field of the FIB, as does the "SetComment" CLI command. The function takes in the 30 character file name, and the comment string, which currently may be up to 80 characters long.

The SetProtection() function takes a long word "protection mask". Only bits 0-3 of this mask are used; they respectively lock out deletion, execution, writing, and reading of their file; a set bit sets the protection, a cleared bit clears the protection. The CLI command "Protect" does the same thing. Finally, the DeleteFile() function accepts a C-string name for a file, which it will attempt to delete, just like the Delete CLI command. All of these functions return TRUE when successful, FALSE otherwise.

WDIR: A PROGRAM EXAMPLE

Listing One contains the WDIR program, which I present here mainly as an example of DOS level programming. This program is in C, and it will compile as written with the Lattice C V3.03 compiler, using either the V1.1 or V1.2 (as of the September 16 release) Amiga systems.

The code should be easily portable to other languages; I've written it to be independent of integer size, and I've used mainly AmigaDOS functions instead of C functions. The program processes one string from the command line, which should be the name of a directory. No argument will default to the current directory. The program allocates a FIB and a lock on the passed directory. It then fills the FIB with information on that directory with the Examine() function.

Once this is complete, the program opens a window at the top of the Amiga's display. Note that this window is opened through CLI, just as any other file. A CLI driven window will always be a window on workbench with a grab bar, sizing gadget, and depth-arranging gadgets.

Anyway, once everything's open, the progam goes into a loop. The ExNext() function uses some context information stored in the FIB to find the next file in that directory. ExNext() will return a 1 when successful, a 0 when unsuccessful. When ExNext() is unsuccessful, the IoErr() function is called. The return value ERROR_NO_MORE_ENTRIES indicates that the last item in that directory was the item described by the current FIB, and so the complete directory has been listed.

The FIB field "fib_DirEntryType" describes the type of file found; a value greater than 0 is a directory, which will be displayed in italics. Any value less than 0 is a normal file. The "fib_FileName" field holds the name of the file as a normal C style string (an array of characters terminated by the ASCII NUL character). The directory loop maintains the "count" variable, which is used to properly display the file names, four per line, until the window is full. At this point, a prompt is displayed, the program waits for a RETURN to be typed, and then we continue.

Finally, the last thing in the loop is a call to the Chk_Abort() function, which returns 0 is everything is OK, non-zero if a ^C has been typed. This allows this dirctory function to act just like the AmigaDOS Dir and List functions, which will all stop for a ^C. The last thing to do is to clean up by deallocating the window and FIB, and then calling the Exit() routine, which again is a standard AmigaDOS exit routine, not the standard C language exit() function.

OTHER AmigaDOS™ FUNCTIONS

I've covered mainly the file-oriented functions of AmigaDOS in this article, those functions that are similar to functions found in many computer DOS systems. There's much more than this available in AmigaDOS.

One of these extra features is asychronous file I/O. Since the Amiga operating system is multitasking, file operations need not take place in a linear ordering, nor should they necessarily cause a program to wait for their completion. Asynchronous I/O is achieved via several data structures called "Packets", which force the I/O to take place in quantized packets with specific requests instead of the streams commonly associated with normal file operations.

Another primary function of the DOS is to manage the Amiga program type "Process". Running programs on the Amiga, of which there may be many, assume two forms, either Tasks or Processes. The ROM Kernal handles Tasks, which are the more basic of programs. The Process structure is built on top of a Task, and a program must be a Process in order to use calls to AmigaDOS. AmigaDOS calls exist to start and stop Processes. A related function is the management of executable code. The AmigaDOS subsystem manages the specific file form for executable code, and there are DOS calls available to load, unload, or execute such segments. Any program that runs from CLI is an example of such executable code.

AmigaDOS™ REFERENCES

I use the Amiga "phone books" for my programming, the version 1.1 series so far. The ROM Kernal Manuals (Volumes I and II) in this section are very complete and provide examples that, while not perfect, are enough to get me going on any problem.

The AmigaDOS Developer's manual is readable, though not as thorough as the RKM. It is a bit vague on the actual form of the data structures and has a few of the structure names either misspelled (fileHandle instead of FileHandle, very significant in C language or other case dependent languages like Modula 2) or not actually stated at all.

Also, the need for special allocation of items (like the FileInfoBlock, which must be word aligned) is not always obvious, and there are no examples in these volumes. The DOS functions are simple, though, and with the help of the Commodore-Amiga supplied include files "libraries/dos.h" and "libraries/dosextens.h", the incomplete documentation wasn't a problem for me. This book also details some of the other AmigaDOS functions, which manage directories, processes, system timing, and the loading of executable code.

The "phone book" series of manuals is no longer published, but they have been replaced by several mass-market books published by Addison-Wesley and Bantam.

The volume describing the things I've discussed in this article is called the AmigaDOS Manual, published by Bantam (I've seen it in many of the local mall bookstores, it should be readily available in most areas, and is well worth the $24.95 if you're serious about any DOS level programming). This volume covers AmigaDOS with a User's Manual section, as well as covering the information contained in the Developer's and Technical Manuals from the phone book series.

Unfortunately, at least the Developer's Manual section seems to be basically verbatim from the Phone Book version, including all of the errors and omissions. Its certainly as complete as the earlier volume, however, and until I see anything better I'd definately recommend it.

Listing One

```
/* WDIR
        Windowed Directory Command
        by Dave Haynie

   This command is designed to get a simple AmigaDOS
directory and display it in an AmigaDOS Window.  This is
intended as an example of using direct DOS calls.  All I/O
is done via AmigaDOS calls instead of using C standard
functions.  This was compiled successfuly under Amiga
Systems 1.1 and 1.2, with the Lattice V3.03 C Compiler.
*/

/*
========================================================== */
/* Includes */

#include <libraries/dosextens.h>

/*==========================================================*/
/* External Declarations */

extern struct FileHandle *Open();    /* Opens a DOS File */
extern void Close();               /* Closes a DOS File */
extern struct FileHandle *Output();   /* Standard Output */
extern long Write();              /* Write to a DOS File */
extern long Read();              /* Read from a DOS File */

extern struct FileLock *Lock();     /* Opens a DOS Lock */
extern void UnLock();              /* Closes a DOS File */
extern BOOL Examine();         /* Gets a FIB from a Lock */
extern BOOL ExNext();     /* Gets next FIB in a directory */

extern long IoErr();           /* Specifics on I/O Error */

extern BOOL Chk_Abort();     /* Explicitly check for ^C */
extern long Enable_Abort; /* Allows explicit ^C checking */

/* ========================================================== */
/* Program variables and constants */

#define LL       80                    /* Lines across */
#define HNUM     4                    /* Names per line */
#define VNUM     (8*HNUM)            /* Names per screen */

struct FileHandle *wind = NULL;/* AmigaDOS Output Window */
struct FileInfoBlock *fib = NULL;    /* File Info Block */
struct FileLock *dir = NULL;/* Locked AmigaDOS directory */

/* ========================================================== */
/* Simple prompt for a return. */

UBYTE go_on(f)
struct FileHandle *f;
{
   char dummy;

   Write(f,"\n\2337mType RETURN To Continue\2330m",30L);
   Read(f,&dummy,1L);
   Write(f,"\233H\233J",4L);
   return (UBYTE) 0;
}

/*
========================================================== */
/* Simple function to open everything cleanly. */

BOOL StartUp(argc,argv)
int argc;
char *argv[];
{
   char *dirname;       /* Selected AmigaDOS directory */
   char wname[108];                    /* Window Name */

 /* First process the command line.  We want exactly one
    directory name specified, but a blank command line
    nicely defaults to the current directory. */

   if (argc == 1)
      dirname = "";
   else if (argc == 2)
      dirname = argv[1];
   else {
      Write(Output(),"Usage: ",7L);

      Write(Output(), argv[0], strlen(argv[0]));
      Write(Output(), " [dirname]\n",11L);
      return FALSE;
   }
```

```
   /* Now we'll need a FIB, a File Information Block.  The
      DOS commands require the FIB to be longword aligned, so
      I AllocMem() it.  */
   fib = (struct FileInfoBlock *) AllocMem(sizeof(struct
      FileInfoBlock),0L);
   if (fib == NULL) return FALSE;

   /* We've got a possible directory name so we'll try to get
      a lock it. Once locked, we'll need the FIB for the
      directory. */
   dir = Lock(dirname,ACCESS_READ);
   if (dir == NULL || !Examine(dir,fib)) {
      Write(Output(),"Invalid directory ",18L);
      Write(Output(),dirname,strlen(dirname));
      Write(Output()," requested\n",11L);
      return FALSE;
   }
   /* Now we create the file name for a console window and
      open it.  The return code is finally based on the
      success of this operation. */
   strcpy(wname,"CON:0/0/640/95/");
   strcat(wname,fib->fib_FileName);
   wind = Open(wname, MODE_OLDFILE);
   return (BOOL) (wind != NULL);
}
/* ========================================================*/
/* Simple function to close everything cleanly. */

void ShutDown()
{
   if (wind) Close(wind);
   if (dir) UnLock(dir);
   if (fib) FreeMem(fib,sizeof(struct FileInfoBlock));
   Exit(RETURN_ERROR);
}

/*======================================================== */
/* The main program */

main(argc,argv)
int argc;
char *argv[];
{
   long len;                  /* File name string length */
   UBYTE count = 0;                        /* Line count */
   /* Make an attempt to open all the necessary things. */
   if (!StartUp(argc,argv)) ShutDown();

   /* Good, we're open.  Now loop for files, until ExNext()
      fails AND IoErr() returns the an appropriate error
      code, or we break out with a ^C. */

   Enable_Abort = 0;
   while (ExNext(dir,fib) != 0 || IoErr() !=
ERROR_NO_MORE_ENTRIES) {
    if (fib->fib_DirEntryType > 0) Write(wind,"\2333m", 3L);
      len = strlen(fib->fib_FileName);
      Write(wind,fib->fib_FileName,len);
      Write(wind,"\2330m                    ",22-len);
      if (++count % HNUM == 0) {
         Write(wind,"\n",1L);
         if (count == VNUM) count = go_on(wind);
      }
      if (Chk_Abort()) {
         Write(Output(),"**BREAK\n",8L);
         ShutDown();
      }
   }=
   /* The last on-screen prompt. */
   if (count != 0) {
      if (count % HNUM != 0) Write(wind,"\n",1L);
      go_on(wind);
   }
   /* And we're done! */
   ShutDown();
}
```

•AC•

"Real Soon Now..."

...words that you will not hear from ASDG

Available NOW!

ASGD experienced some of the same problems other manufacturers have had with production delays. What sets ASDG apart is how we have dealt with our problems. We shipped temporary loaners. In some cases, we have volunteered money back for being behind. In every case we kept our customers completely and accurately informed. ASDG's responsibility to you doesn't end when we cash your check. Rather...that's where it begins.

Mini-Rack-D
a 2 slot totally ZORRO compatible backplane. Powered at +5, -5 and +12 volts. All metal closure. Measures 6"w x 10"h x 10"d. **List $325 with ASDG memory board**. Covered by *Mini-Rack-Buy-Back.*

Mini-Rack-C
a 2 slot ZORRO subset. Powered at +5 volts. All metal enclosure. Measures 6"w x 10"h x 10"d. **List $195 with ASDG memory board**. Covered by *Mini-Rack-Buy-Back.*

Recoverable Ram Disk
Software alternative to RAM:. Files copied to ASDG recoverable ram disk are preserved through resets and crashes. Totally AmigaDOS compatible. Dynamically manages ram disk memory. **FREE with ASDG memory board.**

.5M, 1M and 2M FAST RAM Boards
All ZORRO compatible FAST ram boards. Autoconfiguring. No wait states. .5M (1/2 mbyte) and 1M (1 mbyte) user expandable to 2M (2 mbytes). Optional socketing ($75) makes upgrade completely solderless. 1 year warranty.

		List	Introductory Price	Introductory User Group Price
.5M	1/2 Mbyte of FAST Ram:	$450.00	$395.00	$370.00
1M	1Mbyte of FAST Ram:	$650.00	$575.00	$550.00
2M	2Mbytes of FAST Ram:	$900.00	$795.00	$750.00

ASDG Incorporated

280 River Rd. Suite #54A
Piscataway N.J. 08854
(201) 540-9670

New Jersey residents please add six percent sales tax. ASDG will pay UPS standard delivery in the continental U.S.
Dealer and VMR pricing available. Demonstrations to large user groups.
SYSOPS--Call us for more information about special pricing.

The Amazing C Tutorial

"There is nothing tricky about the work of the preprocessor. "

By John Foust

Like many modern programming languages, C allows the programmer to split a large program into smaller pieces.

This can be done at several levels. A program can be logically divided into functions, a group of instructions called by name. One function can call another, sending it data to be examined. In C programs, the master function is called 'main()'.

The physical source code file containing the program can be split, too. Imagine a C program composed of a thousand lines of code, both data declarations and function declarations. At any one time, the programmer might be adjusting only a few lines of code or data. Yet, to test the program, the entire file must be processed by the C compiler.

The speed of the compile-link-test cycle grows very important to a programmer. When it comes to translating problems to solutions, the human mind moves faster than the compiler. It sometimes seems bugs appear faster than the programmer's mind can imagine.

It is possible to split this thousand-line program into pieces, and separately compile each piece to its own object module. Usually, the linker can join all the object modules in less time than it takes to recompile and link the entire program, if it were present in a single source file.

How would we split this program? This thousand-line program might contain one hundred lines of data declarations and nine hundred lines of function code. A few lines of the data declarations (outside any function declarations) might look like:

```
int base, market;
float pi = 3.14;

struct vector {
    int x;
    int y;
};

struct vector inside = { 3, 2 };
```

Note that the variable 'pi' and the structure 'inside' also contain initialization information. 'pi' is set to 3.14, 'inside.x' is set to 3, 'inside.y' is set to 2.

Remember, without explicit initialization information, static data is initialized to zero by definition while 'base' and 'market' will be set to zero.

Let's suppose the nine hundred lines of function code is separated into thirty functions of about thirty lines each, and that these could be separated into nine logical groups. Each group would be about a hundred lines of source code, containing three functions.

This is a somewhat artificial example, of course. Programs should be written in small modules from the start. A large program should not be composed in a single, large file!

Using our favorite text editor, we could split the large file, moving the function code into separate files, one for each logical group. Each file would be given a filename that reminded us of the logical group's function, such as 'input.c' or 'starchart.c'.

But what about the hundred lines of data declarations? When each function module is compiled, the compiler needs to see the definitions of the data for the module. In other words, if the module wants to manipulate the global variable 'market', the compiler must be told the data type of 'market', in a standard C way.

The most common technique for sharing data declarations is the C preprocessor directive '#include "filename"', which bodily includes the text of the given file when the compiler scans the source code to be compiled. If we move the hundred lines of data declarations to a file called 'datadefs.h', and add the line '#include "datadefs.h"' to every function module, is the problem solved? No.

There is a conflict here: Each module needs to see the data declarations, but only one module is allowed to initialize the data.

continued...

That module's data declaration and initialization looks like:

```
int market, base;
```

In this case, 'base' and 'market' are not initialized explicitly, but are set to zero by default. This module might be said to 'own' these variables.

If they want to reference them, all other modules must prefix the declaration of the variable with the C keyword 'extern':

```
extern int market, base;
```

These 'extern' declarations cannot initialize variables. They only inform the compiler of the data type of these variables. They do not declare storage space, and therefore, cannot initialize space to a certain value.

Just as the code itself is split into modules, '#include' files are often separated by logical function. When compiling a small code module from a multimodule program, compiler spends more time scanning the code for the '#include' files than the source code in the module. Splitting data declarations makes sense, so that the compiler does not deal with extraneous data declarations.

The rules for declaring data and functions are clearly specified in the definition of the C language, but the novice often has trouble implementing the theory. Some aspects of the language definition might seem hell-bent on making the programmer's task more frustrating. With a few simple techniques, the task of splitting a program into pieces is easily accomplished, and avoids the headaches that might seem insurmountable to the beginning C programmer.

For example, a programmer might solve this problem by creating two sets of '#include' files. Both sets would contain declarations, but only one would initialize the variables. The first set would initialize the variables, and the second set would declare those same variables 'extern'. Any module that referenced the global variables would '#include' the header file from the first set. Only one module, the main module, might '#include' all the initializing declarations, and the rest would '#include' the 'extern' declarations.

I have seen both novices and advanced programmers use this technique. This path soon leads to confusion. Ultimately, the programmer will change a declaration in one '#include' file, and forget to change it in the complementary 'extern' file. Or, they will leave off the 'extern' in one module, and quickly realize the mistake when the compiler warns that a variable is declared - and perhaps initialized to zero by default - more than once.

There is an easy solution. It involves a trick of the preprocessor, the part of the compiler that performs simple text substitions on a program file before the code is sent to the compiler proper. This solution merges the two '#include' files described above into a single '#include' file, which eliminates the multiple, parallel '#include' files in the novice solution.

If the problem is stated in this fashion, the answer might occur to you. We want the same '#include' file to contain both declarations and initializations. The initializations should only occur in one module. The 'extern' keyword should be prefixed to the declarations in every other module.

Imagine that all '#include' files have the following lines before any declarations:

```
#ifdef PRIMARY
#define GLOBAL
#else
#define GLOBAL extern
#endif
```

(This only applies to those files that declare and initialize variables. An '#include' file that only contains '#define' directives, for example, would be exempt.)

If you are not familiar with the '#ifdef', '#else', '#endif', and '#define' preprocessor directives, do not be alarmed. Remember the function of the preprocessor. It performs simple text substitutions. 'PRIMARY' and 'GLOBAL' are not variables, they are not part of the program that the 68000 microprocessor will execute in any form. They are strictly commands for the compiler's text scanning functions.

Preprocessor commands are all prefixed by '#'. '#define' is the simplest. This directs the preprocessor to replace all instances of the first supplied symbol with the second symbol.

If the preprocessor encounters this text:

```
#define RATE 3
     ...
     ...
     market = RATE * RATE;
```

then 'market' will be assigned a constant value of 9.

(Compilers will condense constant expressions, such as '3 * 3', into the equivalent value, to prevent the program from performing calculations that can be calculated in advance.)

So the previous example is much like an 'if-else' statement. The '#ifdef' directive controls the preprocessor, instead of program flow. If the preprocessor symbol 'PRIMARY' is '#define'd somewhere in the expanded source code before these five lines are encountered, then '#define GLOBAL' will be scanned by the preprocessor. Otherwise, this "'if' true" block of program text (and possibly more preprocessor directives) will be ignored, until an '#else', '#elseif' or '#endif' is found.

Don't be confused by this first '#define' statement. Perhaps you are more accustomed to seeing '#define' directives with three parts: the '#define', a symbol name, and its replacement text. In this case, there is no replacement text. Any occurrence of 'GLOBAL' will be expanded to nothing by the preprocessor.

In the alternate case, if 'PRIMARY' has not been '#define'd to any value before these lines, then '#define GLOBAL extern' will be scanned. Whenever 'GLOBAL' is encountered in the source code beyond this point, it will be expanded to 'extern'.

So, in the 'main()' module, before the lines of '#include' statements, the line:

```
#define PRIMARY
```

must be present. This line should not be present in any other module. All other modules are free to '#include' all '.h' files. Since '#ifdef PRIMARY' will be interpreted as 'false' in these non-owner modules, the symbol 'GLOBAL' will be expanded to the keyword 'extern', and the declarations will be correct for each module.

An elemental, globally visible variable can be initialized in this fashion:

```
GLOBAL int sunrise
#ifdef PRIMARY
 = 16
#endif
;
```

Note the semicolon, on a line by itself, following the '#endif'. Since it must be present after both the initializer and the simple declaration, it must be outside the '#ifdef'-'#endif' conditional code.

In the 'main()' module, assuming it will "own" all the variables, this code will expand to:

```
int sunrise
= 1
;
```

There is nothing tricky about the work of the preprocessor. All C comments are removed by the preprocessor, along with the preprocessor directives, as in this example. Note the extra blank lines in the expanded source code, the lines where the preprocessor lines were. If your C compiler has an option to save the preprocessed, expanded source code, try it on some of your programs, and compare the output to the original program text.

Structures can be initialized in '#include' files using a similar technique. For example:

```
struct vector {
    int i;
    int j;
};

GLOBAL struct vector inside
#ifdef PRIMARY
= { 3, 2 }
#endif
;
```

continued...

will both declare and initialize this an instance of this 'struct', without contention.

In this scheme, non-owner modules are free to declare their own variables in the normal fashion.

An apology

I must clarify statements I made in the my last C tutorial column. In it, I said "I'm telling you NOT to use C!," because the versions of BASIC on the Amiga are powerful enough for weekend programmers, exploring the machine. Some people appeared to take this personally, as if they once believed I was a great C priest, speaking wise words to the novitiates, and now think less of me because I recommended BASIC over C.

I am not a great fan of BASIC. Like many self-taught programmers, it was my first language, and therefore holds a place in my heart next to my first love, my first merit badge, and my first cool car. BASIC gave me a lot of bad programming habits. Eventually, when the habits became handicaps, it changed my BASIC style. This led me to appreciate the structured style of other languages.

I see little wrong in encouraging people to build a stronger base of programming skills before attempting a more complex or subtle computer language, such as C. Sorry, campers, I am sticking to my guns. Languages are tools; some languages are tools for learning, some are tools for systems programming; none are important enough to spark religious wars.

•AC•

Working with Workbench

"Hey, I didn't even click on any NotePad icon, just that document. Where did notepad come from anyway?"

By Louis A. Mamakos

Danger! Warning! For C programmers and bit pushers. We're going to get down and push the bits around. You should not be afraid of C programming. Others should continue at their own risk.

The AMIGA computer presents a flexible user interface. There is the traditional line oriented interface available using the CLI. This interface is similar in spirit to that available in MS-DOS, CP/M and similar systems.

Of course, there is also the windowing interface using icons, gadgets, menus and the such. It is similar to that available on the Macintosh and the old Xerox workstations. These present the user with a desktop metaphor where he manipulates objects with the mouse.

Having the CLI available makes it easy and fairly simple to port existing programs to run in a terminal emulation window. You invoke the command much the same way as you would a MS-DOS or CP/M program. Your program is loaded, and execution starts. Arguments are available as an array of strings. Thus, a typical C program looks like:

```
main(argc, argv)
        int argc;
        char *argv[];
{
        .
        .
}
```

This is the argc and argv parameters that are passed to the 'main()' function of a C program. Thus, if I type this command at the CLI:

```
1> PROGRAM arg1 foo bar
```

The C program is passed a value of 4 for argc, with 4 arguments; the name of the program (`PROGRAM') as argument 0 (argv[0]), and the 3 other arguments passed as argv[1], argv[2], argv[3]. This works pretty much the same as on other systems.

The only thing is, when I double click on a program icon, there is no command line. Just how do we pass arguments to these programs? Take for example the NotePad program. You can create a note with NotePad, and save it. NotePad creates the file and an icon for it. Subsequently, you can double click the icon for this new file and have NotePad magically fired up to process it. Obviously, something is going on here. Just how does NotePad know to edit that particular file. Hey, I didn't even click on any NotePad icon, just that document. Where did notepad come from anyway?

The little bit of information that ties this all together is the concept of tools and projects. A tool is simply a program, like NotePad or DeluxePaint.

It is code that is loaded and executed and actually does the work. A project is something that you create with a tool. It can be a note or document or graphical image. When a tool creates a project, it can associate a tool with that project. Thus when NotePad creates a document, it associates the NotePad tool with that document. When you double-click on an icon which is project, the AMIGA will attempt to run the tool associated with that project. You can also explicitly indicate a tool to be used for a project by using the extended selection mechanism of the Workbench. This is where you click on more than one icon while holding down one of the SHIFT keys on the keyboard. When the tool is executed, it is passed arguments for the project. This is how NotePad knows what file you want to fool with.

Let's examine how the icons are implemented. If you have a file, call it foo (all sample files are called `foo', didn't you know that?) If the file foo had an icon, it lives in the same directory as the file, and has the string .info appended to it. In this case, the file would be called foo.info. In this file is the imagery for the icon (the little picture), data structures that describe a gadget, stack size, tool types and a default tool. The gadget structure describes what area of the icon you can click in to select it. The stack size is used to allocate the stack space for the tool. The tool types, well, these are interesting things about the project. These are pretty much defined and interpreted by the tool that will process the project. Rather than try to explain what they are used for, I'll give some examples of how they are used in practice later.

As you might expect, the default tool is what tool is invoked to act on the project. Note that this discussion pretty much is restricted to icons of the tool or project type. Drawer and disk icons are somewhat different animals, beyond the scope of this discussion. (Don't you just hate when someone says that? I could of said `Left as an exercise for the reader', but I'll be kind to you instead).

Ok, now that we know pretty much what's going on, how do we C programmers write our programs so that they work in the windowing Workbench environment? Happily, you can write a program that can work well from both the CLI and the Workbench.

I recently did some work on the MicroEMACS text editor available for the AMIGA. I added some stuff to make it work better from the CLI like a real BEEPER and using the console.device handler for better performance. The real interesting stuff has to do with MicroEMACS being able to run from the Workbench, and to create and act on icons for the files that you edit. I'll cover briefly what was changed, and how this is related to programs in general.

First of all, what about these arguments? Just how are they passed to my program when fired up from the Workbench. The situation that I'm going to outline here is for Lattice C. The Manx C version may be different, but the same sort of thing is there, perhaps slightly differently.

As we recall, the 'main()' function of a C program is called with two arguments. The first is the number of command line arguments, and the other is a vector of the actual arguments (as many as specified by the first argument). To specify that the program was invoked from the Workbench, the first argument is set to zero. Recall that when a program is invoked from the CLI, you always get at least one argument (the program name). Thus, the only case where no arguments are passed is when the program is run from the Workbench.

The actual arguments are passed to your program as a message to the new task. The Lattice C startup code grabs this message, and takes care of returning it when your code exits. This startup message is pointed to by the externalized variable WBenchMsg. It is declared as a pointer to a 'struct WBStartup' which looks like this:

```
struct WBStartup {
  struct Message  sm_Message; /* standard message struct */
  struct MsgPort *sm_Process; /* your process descriptor */
  BPTR            sm_Segment; /* descrip for your code */
  LONG          sm_NumArgs; /* # of elements in ArgList */
  char *          sm_ToolWindow; /* descript of window */
  struct WBArg *  sm_ArgList;  /* arguments themselves */
};
```

The interesting thing here is sm_NumArgs which tells us how many arguments are present, and sm_ArgList which are the actual arguments. The other stuff is not important for our discussion. Note that there will be at least one argument always present; the tool that is running. Each of the arguments look like this:

```
struct WBArg {
    BPTR      wa_Lock;        /* a lock descriptor */
    BYTE *    wa_Name; /* a string relative to that lock */
};
```

This is a bit more involved than the arguments that we get from the CLI. For each argument, you are passed the name of the file wa_Name, and a lock on the directory the file is in, wa_Lock. You are passed a lock on the directory because the projects may be in different directories (windows or drawers). To access these files, you have two alternatives. The first is to change into that directory with 'CurrentDir()' before trying to open the file. The other alternative is to recreate the complete path name from wa_Lock and wa_Name. The easiest way is the first alternative; you do much less mucking about the file system. However, in the case of MicroEMACS, it was necessary for reasons that are difficult to explain to convert the wa_Lock/wa_Name pair to an absolute path name.

Let's look at the MicroEMACS 'main()' function:

```
#if     AMIGA
#include        <workbench/workbench.h>
#include        <workbench/startup.h>
#include        <workbench/icon.h>

LONG            IconBase;       /* for icon.library */
char            *getWBname(), *WBstartf;
int             setWBmodes();
void            inittool();
#endif

main(argc, argv)
char    *argv[];
{
        .
        .
        .

#if     AMIGA
  extern struct WBStartup *WBenchMsg; /* Startup message */
        struct WBArg *wbarg;        /* pointer to WB args */
        int narg;                   /* number of WB args */

        if (argc == 0)
        fromWB = 1; /* we're running as son of Workbench */

 if ((IconBase = (LONG) OpenLibrary(ICONNAME, 0))== NULL) {
                fprintf(stderr, "Can't open %s\n", ICONNAME);
                  exit(1);
        }
#endif  AMIGA
   /* init the editor and process the com line arguments */
        .
        .
        .
  /* scan through the com line and get the files to edit */
        for (carg = 1; carg < argc; ++carg) {
                        /* edit in the file */
                        .
                        strcpy(bp->b_bname, argv[carg]);
                        readin(bp->b_bname, TRUE);
                        .
        }
```

```
#if     AMIGA
        if (fromWB) {
      /*
       * if from Workbench, scan the argument list passed
       * to figure out what files we're gonna edit.
       */
                wbarg = WBenchMsg->sm_ArgList;
                narg = WBenchMsg->sm_NumArgs;
                if (narg) {
          inittool(wbarg); /* get startup info from ICON */
           wbarg++;        /* throw away program name */
                        narg--;
                }
                while (narg--) {
                        bp = curbp;
                        makename(bname, wbarg->wa_Name);
                        strcpy(bp->b_bname, bname);
                        strcpy(bp->b_fname,
                    getWBname(wbarg->wa_Name, wbarg->wa_Lock));
           if (readin(bp->b_fname, !viewflag==FALSE)
                        == ABORT) {
                                strcpy(bp->b_bname, "main");
                                 strcpy(bp->b_fname, "");
                        }
                        setWBmodes(bp);
                        wbarg++;
                }       /* while */
        }       /* if (fromWB) */
```

(This example is very paraphrased.) Notice how the variable fromWB is set up to decide if we were invoked from the Workbench or not. The OpenLibrary opens the icon library to do icon related stuff. More on this later. The for loop will process all of the regular arguments, if any, that are in the argv[] vector. Then, we test to see if we were invoked from the Workbench.

If so, we do a special thing with the first argument (which is the tool), followed by processing each of the subsequent Workbench arguments. In this example, the function 'getWBname()' function returns a pointer to the full path name of the argument. 'setWBmodes()' processes file specific modes. These are based upon the ToolType array in the icon. How this is done is demonstrated later.

If you are writing a new program from scratch, you will probably want to change into the directory specified by the lock, rather than expanding the complete path name. You would do something like this:

```
struct WBStartup *WBenchMsg;

main(argc, argv)
        int argc; char **argv;
{
        int fromWB;

        fromWB = (argc == 0);

        if (!fromWB) {
                /* from CLI */
                for (i=1; i < argc; ++i) {
                        doarg(argv[i]);
                }
        } else {
                /* from WB */
                BPTR olddir;
                int narg;

               for (i = 1; i < WBenchMsg->sm_NumArgs; i++) {
                    /* change in to that arg's directory */
    olddir = CurrentDir(WBenchMsg->sm_ArgList[i].wa_Lock);

                    /* do something with it */
                    doarg(WBenchMsg->sm_ArgList[i].wa_Name);

                    /* restore current directory */
                    (void) CurrentDir(olddir);
                }
        }
}

doarg(file)
        char *file;
{
        FILE *fp;

        fp = fopen(file, "r");
        /* do something with it */
        fclose(fp);
}
```

Again, this is just a skeletal program, but you'll get the idea.

Now that we've got an idea how to get the arguments from our program running under the Workbench, let's examine the icons themselves. As I alluded earlier the icons themselves can have some useful information in them, namely the ToolType array. The ToolType array is a way of parameterizing a project or tool outside of the file. For example, in MicroEMACS the editing modes are stored in one of the ToolType entries. The NotePad program stores the window size, default fonts and other settings of the document that you were editing.

To examine the ToolType array, the easiest way is to select (but not run; only one click), and then select Info from the Workbench menu. This will put up a window and display most of the good stuff in the icon. Commodore-Amiga doesn't document the format of the icon file stored on the disk. Happily, they have supplied a set of standardized library routines to access the icons. It is important to note that the only documented interface and description of the icons is when the icon has been read into memory using the support routines in the icon.library library. This is the purpose of doing the OpenLibrary("icon.library", 0)} in the MicroEMACS 'main()' function. The three basic functions are 'GetDiskObject()', 'PutDiskObject()' and 'FreeDiskObject()'.

'GetDiskObject()' is called with the name of the icon to get. Note that you do not supply the .info suffix for the file. When it finds the icon, it allocates some storage, and returns a

continued...

pointer to a 'struct DiskObject'. This is the format that is documented by Commodore-Amiga for dealing with icons.

'FreeDiskObject()' is used to free the storage allocated by 'GetDiskObject()' above. For every 'GetDiskObject()' you should have a corresponding 'FreeDiskObject()'. It takes a single parameter, which is the 'struct DiskObject' pointer returned by 'GetDiskObject()'.

'PutDiskObject()' is used to create or update an icon. It takes two parameters; the name of the icon and a pointer to a 'struct DiskObject' set up previously to describe the characteristics of the icon. You can create your own 'DiskObject' structure, or use one that is returned by 'GetDiskObject()'.

Let's look at some code that uses these functions to create and handle icons. The first function that we'll look at is one that is used by MicroEMACS to create an icon for a file that doesn't already have one.

```
#define G_Width    21
#define G_Height   33

/* Image Data */
UWORD ProjObjData[] = {
/* Bit Plane #0 */
   0x0000,0x07ff,
   0x0000,0x07ff,

   0x0000,0x07ff,

/* Bit Plane #1 */
   0x0000,0x07ff,
   0x0020,0x07ff,

   0x0000,0x07ff
};
```

The first thing that we see is a bitmap definition of the icon image. Most of it has been deleted since it's pretty boring. (For your infomation, this definition was produced from a DeluxePaint brush file by a program called 'gi').

```
struct Image ProjObjImage = {
        0, 0,             /* left edge, top edge */
        G_Width, G_Height, /* width, height - pixel size */
        2,                /* depth - pixel size */
        &ProjObjData[0],/* pointer to image data */
        3,0,              /* plane pick, plane on/off */
        NULL,             /* next image */
};
```

The next thing in line is 'struct Image' structure. This is an Intuition data structure that defines the form of the image data (size, bit planes, etc). This is tied to the actual image data. 'G_Width' and 'G_Height' are the size in pixels of the image data.

```
struct DiskObject ProjObj = {
        WB_DISKMAGIC, WB_DISKVERSION,
        {
                /* gadget structure */
                NULL,                          /* next gadget */
                0, 0,              /* left edge, top edge */
                G_Width, G_Height, /* wid, hht - hit box */
                GADGHBOX|GADGIMAGE,            /* flags */
                RELVERIFY|GADGIMMEDIATE,/* activation */
                BOOLGADGET,                    /* gadget type */
                (APTR) &ProjObjImage,    /* gadget render */
                NULL, NULL,/* select render, gadget text */
                0, 0, 0, 0, /* mutual exclude, spec info,
                               gadgetid, userdata */
        },
        WBPROJECT,    /* object type (project, not tool) */
        "SYS:Utilities/Emacs",         /* default tool */
        &ToolTypes[0],                 /* tooltypes */
        NO_ICON_POSITION, NO_ICON_POSITION,
                                       /* icon pos unspec */
        NULL, NULL,       /* drawer data, tool window */
        0                              /* stack size */
};
```

Ah, this is the interesting part. Here we've defined the 'struct DiskObject' called 'ProjObj' ourselves. The first two constants, WB_DISKMAGIC and WB_DISKVERSION are `magic' numbers defined in the intuition include file. This is used as a check to make sure that intuition is looking at a DiskObject structure.

The first component we see is another structure, a 'Gadget' structure. This defines the appearance of the icon, what part of it is sensitive to being clicked on, and how to highlight the image. It turns out that in this case, the `hit box' or the part sensitive to clicks is exactly the same size as the image, namely 'G_Width' by 'G_Height' pixels. We can change the highlighting from HIGHBOX to HIGHIMAGE and supply another image structure to cause the icon to display an alternate image when selected, rather than the normal highlighting method.

The next thing that we see is the constant WBPROJECT which identifies this icon as a project, rather than a tool. The default tool is specified next, which in this case is SYS:Utilities/Emacs. The ToolTypes for this file are written next. By default, they are initialized at compile time to:

```
UBYTE *ToolTypes[] = {
        "FILETYPE=MicroEMACS.document|text",
        "MODES=",
        NULL
};
```

Note that the MODES tooltype is modified in the actual code. The list of ToolTypes is terminated by a NULL pointer. The position of the icon (the X and Y coordinates of the icon in the window) is defined next. When creating a new icon, you

can supply the special value NO_ICON_POSITION which will cause Intuition to find a good place to display the icon when you open the drawer that the icon is stored in.

These are set to absolute values when you use Snapshot from the Workbench menu. Following that is a field which is used when the icon is a drawer, which is unused, as is the tool window field. The last field in the structure is the stack size. This will determine the size of the stack for the tool which will be invoked to process this project. By setting it to zero, we allow the stack size in the tool's icon or the system default determine the size.

And now, some code:

```
/*
 *  Write a default icon out for the specified file.
 */

putdeficon(bp)
        BUFFER *bp;
{
        register char *cp;

        if (DefaultTool)    /* if we've got a better idea */
                ProjObj.do_DefaultTool = DefaultTool;

    /* borrow fibuf[] above to construct tooltype thingy */
        strcpy(fibuf, "MODES=");
        if (bp->b_mode & MDWRAP)
                strcat(fibuf, "WRAP|");
        if (bp->b_mode & MDCMOD)
                strcat(fibuf, "CMODE|");
        if (bp->b_mode & MDEXACT)
                strcat(fibuf, "EXACT|");
        if (bp->b_mode & MDOVER)
                strcat(fibuf, "OVER|");

        cp = fibuf;
        while (*cp)

                ++cp;
        if (cp != fibuf)
                --cp;               /* back up to last character */
        if (*cp == '|')
                *cp = '\0';

        ToolTypes[1] = fibuf;     /* use our mode string */
        return(PutDiskObject(bp->b_fname, &ProjObj));
}
```

The 'putdeficon()' function takes a pointer to a data structure used in MicroEMACS called a BUFFER. All that you need to know is that there is a field in this structure, called b_fname which is the name of the file, and another field called b_modes which is a bit mask of editing modes. If we've discovered another default tool (by DefaultTool being non-zero), then we replace the compiled in default tool by that one. Next we construct a ToolType entry called MODES.

This will be composed of editing modes that are currently active for this BUFFER. We list the modes as strings separated by the vertical bar character. This is done for reasons that will soon be clear. Once we've constructed the MODES ToolType, we replace the compiled null MODES ToolType with the one that we just built. Then we write the ProjObj DiskObject structure out to the file bp->b_fname specified in the BUFFER structure, and presto! We've got a new icon on the disk.

Note that we don't have to call 'FreeDiskObject()' since the DiskObject was not allocated by 'GetDiskObject()'.

Let's look at a function that processes an exiting icon (or DiskObject). In the 'setWBmodes()' function, the DiskObject has already been read when the file off disk, and a pointer to the DiskObject structure is stored in the BUFFER structure.


```
/*
 *  This func sets the buffer specific modes according to
 * the tooltype array that might be in ICON of the file.
 */
void
setWBmodes(bp)
        BUFFER *bp;
{
        char *tt;
        struct DiskObject *dskob;

  /*
   * this proc is to check the tooltypes array for the
  * edit modes to be set for this file. Do nothing for now.
   */
        if (bp->b_DiskObject == NULL)
                return;  /* if no ICON for file, skip it */

        dskob = (struct DiskObject *) bp->b_DiskObject;
  if ((tt = FindToolType( dskob->do_ToolTypes, "MODES" ) )
                == NULL)
                return;      /* if no MODES tool type */

/*
 * Check each of the modes and set the appropriate flags if
 *  found
 */
        if (MatchToolValue(tt, "WRAP"))
                bp->b_mode |= MDWRAP;

        if (MatchToolValue(tt, "CMODE"))
                bp->b_mode |= MDCMOD;

        if (MatchToolValue(tt, "EXACT"))
                bp->b_mode |= MDEXACT;

        if (MatchToolValue(tt, "OVER"))
                bp->b_mode |= MDOVER;
}
```

First of all, we don't see the 'FreeDiskObject()' call here; that lives in some code elsewhere which frees all of the DiskObjects when MicroEMACS terminated.

We see for the first time the use of the function 'FindToolType()'. This searches the ToolTypes array for a specified ToolType. In this case, we're looking for the

continued...

MODES ToolType, presumably created by the function we saw earlier. If one doesn't exist, we just return. If we did find the MODES ToolType, we test it for specific values. Here we use another function in icon.library, called 'MatchToolValue().' It searches a given ToolType, in this case MODES, for a target. The target is delimited from other targets by the vertical bar character. This is why we used it before. If we find a particular mode, then we set the corresponding mode bit in the BUFFER structure.

Finally, we look at an example which combines a number of these support routines. It is invoked during the MicroEMACS initialization process with a pointer to the WBArg structure for the MicroEMACS tool rather than for a project that we are editing. This combines a number of the techniques mentioned before.

```
/*
 *  This func is called at init time to process the
 *  tooltypes of the emacs tool.  This is to extract some
 *  useful info, such as the default project for the icons
 *  that we create.
 */

void
inittool(wbarg)
        struct WBArg *wbarg;
{
        BPTR olddir;
        struct DiskObject *diskobj;
        register char **tt;

        if (wbarg == NULL)
                return;
        olddir = CurrentDir(wbarg->wa_Lock);
        diskobj = GetDiskObject(wbarg->wa_Name);
        if (diskobj == NULL) {
                (void) CurrentDir(olddir);
                return;
        }
        DefaultTool = FindToolType(diskobj->do_ToolTypes,
                                   "DEFPROJTOOL");
        if (DefaultTool)
                DefaultTool = newstr(DefaultTool);
        /* other stuff can be searched for in here... */
        FreeDiskObject(diskobj);
        (void) CurrentDir(olddir);
        return;
}
```

First, we need to change into the directory where the tool lives; and get the disk object. Then we check for a ToolType called DEFPROJTOOL. If thisToolType exists, then the value of it is used as the default tool for the projects created. Thus, the user can override the compiled in constant "SYS:Utilities/Emacs" above by adding a ToolType to the MicroEMACS tool with Info. Note that we call 'newstr()' to make a copy of the string because the pointer returned by FindToolType is in the DiskObject structure which is freed later on. Implementation of 'newstr()' is left to the user. (Ha!) Finally, the DiskObject structure is released, and we change the current directory back to where we started from.

Well, that's all there is to it. Please remember, the code fragments included above are just that; fragments. Include files are not listed, and variable declarations not present. Hopefully more programmers will write applications that take advantage of the Workbench window environment to the fullest extent possible. A little bit of effort can go a long way.

•AC•

The AMICUS Network

By John Foust

The World of Commodore, Amiga Desktop Publishing, New Amicus Disk and More...

World of Commodore
In early December, the World of Commodore show was held outside Toronto, Canada. Over 35,000 Commodore fans stepped through the turnstiles at this four-day show.

The long-awaited Sidecar was selling to the general public. The Canadian dollar price at the show wavered between $1300 and $1100, which puts the US price in the $795 ballpark, after considering Canadian exchange rates and customs fees. At press time, several sources claim some US dealers have sold their demonstrator Sidecars for around $600. The Sidecar is also being sold in Europe.

The show is not exclusively Amiga products, but covers the entire Commodore line. The newly-designed Commodore 64-C was on display, along with many Commodore 128 products. Commodore also showed the PC-10 and PC-20 IBM compatible machines, which will be introduced to US markets this month, after enjoying sales success in Europe for quite some time.

Many of the booths were hosted by Amiga dealers, the largest being Phase Four from Calgary, Canada. They subdivided their booth into a dozen or so mini-booths for mostly Amiga-related products, including space for NewTek, who showed the Digi-Paint HAM editor, and new Digi-View software.

Several vendors showed other long-awaited Amiga hardware products. Xebec and C Ltd. showed hard disks, while some dealer booths displayed MicroForge and Byte-by-Byte expansion boxes. Genlocks were attached to Amigas in other booths, such as the Mimetics display. Their booth is always surrounded by crowds gawking at the MTV-style videos and great synthesizer music.

Aegis Development showed Sonix, the latest incarnation of Musicraft. Sonix adds music printing and MIDI capabilities. Aegis also showed improved versions of their vector-based computer-aided drawing programs, such as Aegis Draw Plus.

Amiga desktop publishing
Commodore officials made a strange pronouncement at the Monterey developer conference, in early November. Out of the blue, they declared the Amiga a perfect desktop publishing computer, and claimed its future would be built on that software base. It is said that this strange edict is due to the influence of a single person in upper management at Commodore.

Of course, this imposition left many people shaking their heads. It puzzled many to hear desktop publishing was the saviour of the machine, when no such software existed.

Commodore might have known about the progress of Gold Disk and their PageSetter program at the time of the developer conference, and certainly knew about Publish!, a desktop program announced but not yet on the market. Why make a sweeping prediction like that, at a conference of people writing everything else but desktop publishing software?

If this is true, it will be a battle against the Goliath of Apple Computer. The Macintosh stands tall over the desktop publishing market, and Amiga would have to find nourishment in that shadow. There are other reasons why the Amiga and today's Amiga desktop publishing programs don't stand a chance against Apple.

PageSetter
The World of Commodore show marked the debut of PageSetter, the first desktop publishing program for the Amiga. It comes from a team of talented programmers in a city outside Toronto.

PageSetter's output is destined for nine-pin dot matrix printers. It will be a great program for home-crafted newsletters, but it just doesn't have the quality of a laser printer-oriented program. Until this point, I imagine there are a lot of Amiga user groups using Commodore 64s to produce their newletters, because no such software existed for the Amiga.

Publish! was the first announced Amiga desktop publishing software. Brown-Wagh distributed leaflets announcing the program at the West Coast Commodore Association meeting this fall in Los Angeles.

To me, it isn't fair to compare the two at this time, since PageSetter is being sold now, and Publish! is not. But based on the demonstrations at World of Commodore, most onlookers said Publish! doesn't compare to PageSetter.

I also saw a very preliminary version of another desktop program, called City Desk, from SunRize Industries, the makers of the Perfect Sound audio digitizer.

PageSetter is well made. I get the feeling the PageSetter programmers knew their program would be a crossroads of many existing Amiga programs, and therefore, they made data transfer as flexible as possible. PageSetter has an integrated text and graphics editors. The text editor accepts ASCII, Textcraft and Scribble documents without question. The graphics editor loads color IFF pictures of any resolution, and creates a black-and-white version of it, using different dot patterns. You can import color pictures, and print them in black-and-white. Smart!

Desktop limits

PageSetter is a nice program. I have been tinkering with it since the World of Commodore show. It will sell well in the Amiga market. I would like to explain why this incarnation of PageSetter will not fulfill Commodore's dream.

For output, PageSetter uses the printer drivers present in the Amiga. This is a both a blessing and a hindrance. First, the nice part. The Amiga's system of universal printer drivers means PageSetter can instantly address almost every printer connected to an Amiga today. If your printer can print Amiga graphics today, then PageSetter will work with your printer. The HP LaserJet laser printer has an Amiga printer driver, so PageSetter produces its best output on that.

The bad news? These same printer drivers have limitations, and the PageSetter program is restricted by them.

Bit-maps

Bit-maps are the most common method of producing images by computer. Perhaps you have heard that term before. A 'bit-map' is a grid of dots. Printed images are created by printing or not printing a dot at a particular spot in the picture.

A dot-matrix printer prints a fixed collection of bit maps that look like the letters of the alphabet. Most dot-matrix printers can print graphic images, as well as letters. To print a graphic image, the host computer sends the printer thousands of bytes of data, representing the image, in the form of a bit-map from computer memory. Instead of printing letters, the printer is asked to print graphics images.

The Amiga screen is a large bit-map. Each character on the Amiga screen is composed of small dots. Screen-dump programs send the screen's bit-map directly to the printer.

The burden of the bit-map

The apparent quality of a bit-map image depends on the density of the dots. Normal Amiga text screens are a grid or bit-map of 320 by 400 dots, rather coarse by most printing standards. For example, this magazine is produced on an Apple LaserWriter laser printer, which makes 300 dots per inch. Today's top-of-the-line dot-matrix printers have about 200 dots per inch in the horizontal direction, but much less in the vertical. A future modelo of the Apple LaserWriter will have 450 dots per inch. The best commercial typesetting machines have more than a thousand dots per inch.

PageSetter has a maximum dot density limit, imposed by both the programmers and today's dot-matrix printers. If PageSetter increased the density of dots it produces, it would require more memory, since every dot is stored as a bit in memory.

This problem has already surfaced in PageSetter. You can print your document from within the program itself, or you can use a stand-alone printer program that could run in the background, while you worked with another program. When you print within PageSetter, memory is in short supply, so the program can only create a small portion of the bit-map in memory at a time. After it sends this part of the bit-map to the printer, it must send a reset code to the printer. The reset code often moves the paper slightly, as if you just cycled the printer's power switch. Some people complain that this introduces a white line in their printouts.

I must stress, this is not strictly a fault of PageSetter, but of imprecise printers. The stand-alone printer program can create the entire bit-map at once, and the white-line problem goes away. In either program, if dumping to a laser printer, the problem does not exist.

You might guess this is a losing game - that output devices will always be out-racing the programs that feed them. This is true. Imagine a desktop program that is asked to draw a straight line, across a page, on the diagonal. The line will be composed of a stair-step effect of dots, when viewed closely. If the output device has dense enough dots, and the program can output a dense enough bit-map of dots, then the stair-stepping won't be visible. If either falls short in the density of the bit-map, then the 'jaggies' will appear.

This might be the reason you are dissatisfied with the output from the NotePad program on Workbench. The characters are composed of coarse dots that might look nice on the low-density bit-map of the screen, but look awful when printed on the higher-density bit-map of your dot-matrix printer.

(Another reason involves the way the printer driver translates a bit-map to the printed page. The graphics calculations involved include division, and computers can accumulate errors in division. These inaccuracies lead to occasional disproportionate lines in a graphics dump.)

The Postscript solution

There is another solution. It is used in the Apple LaserWriter. Instead of forcing the host computer to create the bit-map, why not make the laser printer smarter, and let it draw the line to its best density? By sending the printer higher-level instructions, instead of a raw bit-map, the host computer is freed of excessive memory requirements, and the output is guaranteed to be the finest possible.

These higher-level instructions are called 'page description languages.' The most common today is called Postscript. Postscript is truly another computer language. Each page to be printed is described in Postscript's terms, and this program is sent to the laser printer. This program might include the text of this article, instructions about which fonts to use, border information, bit-map screen graphics.

The Postscript description of a page of graphics and text is sent to the LaserWriter in human-readable form. Graphics are dumped in hexadecimal, so that part would look like streams of undecipherable data, but the text portions would be directly readable, surrounded by formatting and font selection commands.

Often, the LaserWriter has more memory than the Macintosh driving it, since the burden of the bit-map has shifted to the printer's shoulders. A Postscript language interpreter program is always running inside the LaserWriter. When a new program is sent to the LaserWriter, the interpreter program translates commands to select text fonts, place graphic images, format text, and draw lines, for example. All these commands are eventually translated to a bit-mapped image, which is transferred to the paper.

Until Amiga desktop publishing programs are sending Postscript commmands, and not bitmaps, we will be behind the times.

Are there any barriers to Amiga programs using Postscript? No, none whatsoever. There are many Postscript compatible laser printers in the market today. The Apple LaserWriter is the most common. Even the LaserWriter will work well with the Amiga, since it uses a simple RS-232 serial interface to receive commands.

Both PageSetter and Publish! representatives claim their programs will have Postscript upgrades a few months after introduction.

Postscript engines

Since Postscript is just a computer language, why couldn't it execute inside the host computer, instead of inside the printer? The LaserWriter has a 68000 microprocessor inside, as does the Amiga, Atari St and Macintosh. The LaserWriter's Postscript program is already written in 68000 code, in ROM memory inside the printer. Chances are they wrote most of the software in a high-level language such as C.

Given enough memory - say four to six megabytes standard - a future inexpensive Amiga or Atari machine could be coupled with a cheap laser printer engine that does not under-stand Postscript itself. Along with the desktop

publishing software, the computer could be loaded with a Postscript driver program, similar to today's printer drivers. This frees the desktop program of the burden of the bit-map, and shifts it to the driver. The cheap laser printer only understands raw bit-maps, like the dot-matrix printer.

This strategy depends on the current price gap between smart, Postscript-driven laser printers, and cheap, bit-map driven laser printers. Cheap laser printers are often called 'engines', since they do little more than print bit-maps. With slightly different hardware, a laser printer engine can become a smart copying machine, or a facsimile machine. Today, a smart laser printer is at least double the cost of a dumb engine.

If the price difference stays great, then you can bet Amiga and Atari will aim at capturing the low-end desktop publishing market. Their memory rich, inexpensive but powerful machines will do well at manipulating the large bit-maps that describe a page of text. Of course, the Amiga's custom blitter chip will work wonders with page bit-maps, as it now does with screen graphic bit-maps.

World of Commodore, Con't.
The World of Commodore show looked so small when I entered the floor. I looked to the right and the left, expecting a passageway to a larger room, something COMDEX-sized. At first, I thought I had been fooled by the promotional videos I saw at the West Coast Commodore Association show in September. They used a wide-angle lens, I thought, to make the show floor look much bigger than it was.

It is nice to see the same people at every show. Many Amiga developers are one- and two-person businesses. Half the company might be on the road demonstrating the product!

It is nice to meet the same old friends in different cities. It means you can always find someone for dinner or dancing after the show. It is nice to meet new people in each city. Travelling has increased my memory for names and faces. It is always exciting to talk with an enthusiastic Amiga supporter, someone who organizes a local Amiga user group. In Toronto, I met a man who holds the group meetings in his own home, down in the bayou of Louisiana.

Only the press and speakers had badges at World of Commodore. Badge inspection is a common game at most computer conferences. People's eyes are often locked at eye-level, quickly scanning the names on the badges of the people walking through the aisles, to catch a glimpse of someone from Microsoft, IBM, or Borland. I am always scanning to spot other computer journalists, who are marked with a specially colored ribbon.

The ribbon serves as a warning to people speaking about non-public information. Some folks get real quiet when they see a press ribbon approach. Conversation stops until they are sure you are not from an important magazine. In one case, an entire table of people moved away from me, when I sat down to eat lunch. My companion and I both wore press badges. It is enough to make one paranoid.

After scrambling for a seat on a courtesy bus at COMDEX Fall, I sat down next to Brett Glass, a organizer of the BADGE Amiga user group in California. I have seen many of his messages on Usenet and the Well, but I had never met him in person. If I wasn't scanning badges, I would never had met him. At COMDEX, Amiga enthusiasts were few and far between.

It is always surprising to meet an online friend in person for the first time. An 'online friend' is someone from a computer network. Using my modem and Amiga, I talk with dozens of people a night. I talk with them much more than my friends here in my city. It is hard to explain that I spend so much time with people I have never met.

These online friends are many, many miles away. On a given night in electronic conference, I might be chatting with people in Los Angeles, Maryland, Florida, Illinois and London. I might never meet them in person, but I know them as friends. Meeting people from far away almost gives the same feeling as travelling; you find out what makes your life different from theirs, and what might be the same.

At the AmigaWorld reception at COMDEX, I talked with someone wearing a moose hat (don't ask) for many minutes. (At a slow party, always follow the people wearing ridiculous hats.) As that party died, we joined a group of Amiga people heading for the Borland party. I wasn't wearing a tie, so Borland's bouncers gave me a very ugly tie to place around

the collar of my polo shirt. Again, I sat down next to the guy in the moose hat. As that party ended, we finally exchanged business cards. It turns out we had met online many times before. We also have many mutual online friends.

With the miracles of telecommunication, friendships are no longer bound by physical location, but electronic watering holes. On that note, I hope you enjoy this telecommunications issue. Telecom is not only a fast track to the Amiga community, it is a path to a community of the future.

AMICUS Network news

When the Amiga was young, I dreamt of assisting the development of a national Amiga user group. I created the AMICUS Network, and started out with a small newsletter, and the AMICUS public domain disks. With the advent of Amazing Computing, this column supplanted that medium, and the AMICUS disks took on a life of their own.

On People Link, I talked with Jim Meyer, a principal in the Hudson Valley Amiga group, and he revived my hopes for a national Amiga user group. At our local user group meetings, I talked with members who wanted to exchange newletters with other Amiga groups. I lent my collection of newletters to the group librarian, so he could make copies for the group lending library.

Meyer and I also talked about exchanging newsletters on disk. National computer networks would allow instant access and wide distribution, but disk exchanges might be more practical for exchanging large amounts of information. In cooperation with the publishers of Scanlines, the Los Angeles area Amiga user group newletter, there will be a system for exchanging newletters and programs on disk between user groups around the nation.

So a national Amiga group is in the works, named the AMICUS Network. If you are a leader in an Amiga group, please get in contact with us on People Link. His ID is 'NY*JIM', mine is 'AMICUS'. So far, the plans for the group include newsletter exchanges, disk exchanges, and databases of reviews and product information.

Every Sunday night, user group members are welcome to join in an online conference. The latest plans for the group will be discussed. Also, the People Link Amiga Zone now has a special section devoted to this new national group, where user group newletter files will be uploaded and available to all People Link users. Once things get off the ground, we hope to expand to other networks, to reach as many people as possible.

Too Late Now!

The phrase "Real Soon Now" - abbreviated "RSN" - is common in the computer world. It is a favorite among marketing-types promising future software releases. It has become both a joke and a serious statement.

Taken tongue in cheek, it expresses typical computer buyer cynicism. Marketers continue to use it, well aware of the common definition, somehow hoping that their use of such a popular cynicism will endear them to the rubes in the aisles, and sell more product, when it finally arrives.

I heard another acronym - "*TLN.*" Some products are announced well in advance of their expected release date, then fall victim to standard production delays, and finally limp to market, far behind competitors, or far behind the initial surge of consumer enthusiasm for the product. This phenomena should be familiar to Amiga owners, in the form of Sidecar, Genlock and friends.

"TLN" stands for "Too Late Now" - as in this imagined conversation:

Person 1: "They are selling Sidecars at the local Amiga dealer!"
Person 2: "Too Late Now."

New AMICUS disks

There are two new AMICUS disks. AMICUS 15 and 16 are collections of the newest public domain software.

AMICUS 15:

The C programs include:

'pr' is a file printing utility, which can print files in the background, and with line numbers and control character filtering.

'fm' displays a chart of the blocks allocated on a disk. You can enter a file name, and see which part of the disk is used to store that file.

'Ask' asks a question in an 'execute' file, and returns an error code to control the execution in that batch file. For example, you could ask whether to copy some files to the RAM: disk, in your 'startup-sequence' on your Workbench disk. Or it could ask to run a certain program, or exit to the CLI.

'Stat' is an enhanced version of the AmigaDOS 'status' command. It tells the priority, address and the command line executed for each program running, plus its current directory.

'Dissolve' is the random-dot dissolve demo described in a recent issue of Dr. Dobbs Journal of Software Tools. It displays an IFF picture slowly, dot by dot, in a random fashion. Slowly, the complete image is built up.

'PopCLI2' is the most recent version of the program from the Software Distillery that can invoke a new CLI window at any time, at the press of a key. This is handy for doing CLI work after forgetting to start a CLI window before running another program.

The executable programs include:

'Form', a file formatting program that works through the Amiga printer driver to select the many different print styles available. By embedding slash commands within a text file, and printing it with 'Form', it could print certain words in boldface, for example. These commands look like '\bThis is bold\b' to print those three words in boldface.

'DiskCat' catalogs disks. It maintains and sorts lists of the files on disks, and can merge and selectively sort these lists. Perhaps a future AMICUS disk will contain an electronic catalog of all the AMICUS and Fred Fish disks, the database created by a program such as this.

'PSound' is the sampled sound editor and recorder from SunRize Industries. They make an Amiga sound sampler box called Perfect Sound. This software comes with it. Amazingly, it works with both the FutureSound sound sampler, and their own hardware. The Perfect Sound software is much easier to use than the FutureSound software, although it can't sample at the maximum rate of the FutureSound box. SunRize also markets disks of sampled sounds that you can incorporate into other programs.

'Iconmaker' makes icons for most programs that do not have them. It provides a neat interface to the creation and selection of all the icon types. Please note that some programs MUST be run from the CLI, and that 'iconmaker' cannot make icons for all programs.

'Fractals' draws marvelous fractal seascapes and mountainscapes. Perhaps you have heard about 'fractals'. This term is used to describe a method of other methods, computer-generated objects look very computer-generated, very non-lifelike. Fractals are a precise way of creating non-regular objects. Creative use of fractal formulas can create realistic looking mountain ranges and plants.

'3D Breakout' is a bizarre version of an old computer game favorite. Get out your red-and-blue lensed glasses for this one. It displays a view looking into a room, and the walls of the room are tiled with the bricks normally seen in a computer 'breakout' game. The floor holds a paddle, and the mouse moves the paddle, which can deflect the ball to break the bricks on the walls. All four walls! With 3D glasses, it appears as if you are looking into the room, with the paddle moving in three dimensions.

'AmigaMonitor' is a window into the intimate system resources inside your Amiga. This is a technically oriented program that displays lists of open files, memory use, tasks, devices and ports in use in the Amiga operating system.

'Cosmoroids' is a version of the 'asteroids' video game for the Amiga.

'Sizzlers' is a high resolution graphics demo written in Modula 2.

The texts on AMICUS 15 include:
'ansi.txt' explains the escape sequences that the CON: device responds to. The CON: device is the CLI window, in essence. It responds to special sequences of characters, each beginning with the character called 'escape'. For example, there are escape sequences to clear the screen, to erase only portions of the screen, to change to italics or boldface, or to change the text color.

'FKey' comes all the way from Australia. It includes an IFF picture that is a template for making strips of paper to sit in the tray at the top of the Amiga keyboard.

'Spawn' is a programmer's document from Commodore Amiga, describing ways to use the Amiga's multitasking capabilities in your own programs.

The AmigaBasic programs on 15 include:
'Grids' is a program to draw sound waveforms, and hear them played. 'Light' is a version of the Tron light-cycle video game. 'MigaSol' is a game of solitaire. 'Stats' is a program to calculate batting averages for up to 20 baseball players, over several games.

As the program itself says, 'Money' is a game where you "try to grab all the bags of money that you can." The bags of money fall from the sky, and you move your player under them as they fall. In the next phase of the game, tax collectors fall from the sky, and you avoid them. The Amiga gives people the creative edge, right?

AMICUS 15 also includes two beautiful IFF pictures, of the enemy walkers from the ice planet in Star Wars, and a picture of a cheetah.

AMICUS 16
AMICUS 16 contains a by-now famous demo of HAM animation on the Amiga, the juggler demo by Eric Graham. This demo is astounding. It shows a robot juggler bouncing three mirrored balls, with sound effects. Twenty-four frames of HAM animation are flipped quickly to produce this image. You control the speed of the juggling. The author's documentation hints that this program might someday be available as a product.

This disk has two IFF pictures, parodies of the covers of Amiga World and Amazing Computing magazines.

The C programs include:
'Inputhandler' is an example of making an input handler. Such a program converts a keystroke to another set of keystrokes, before the keystrokes are sent to an application program. Imagine your keystrokes coming from the keyboard in a pipe, to the program. The input handler converts certain keys to other keys, as they flow through the pipe. This example converts a set of keystrokes that are useful in the editor TxED.

'FileZap3' is a binary file editing program. With this program, it is sometimes possible to change the text string messages in your favorite program, or to change the files and directories it uses by default.

'ShowPrint' displays an IFF picture, and then prints it. It allows you to abort the printing before the picture is finished.

'Gen' is a program that indexes and retrieves the C structures and variables declared in the vast Amiga include file system.

The executable programs include:
'FixHunk2' is a must-have program for anyone with a memory expansion. Some programs do not work well in expanded memory. It is no fault of the Amiga; the blame rests solely on the programmer! Fixhunk can repair an executable program file, so that it has a much better chance of running in expanded memory. Some memory board manufacturers include a copy of FixHunk on the disk that comes with the hardware.

'*ms2smus*' converts Music Studio song files to the IFF standard 'SMUS' format. I have heard this program might have a few bugs, especially in regards to very long songs, but it works in most cases. It fills a niche, so let us hope the bugs are fixed in a future version.

'Missile' is an Amiga version of the 'Missile Command' video game, complete with sound effects.

This disk also contains several files of scenarios for Amiga Flight Simulator II. By putting one of these seven files on a blank disk, and inserting it in the drive after performing a special command in this game, a number of interesting locations are preset into the Flight Simulator program. For example, one scenario places your plane on Alcatraz, while another puts you in Central Park.

Consumer Electronics Show
Commodore will have a booth at the Consumer Electronics Show in Las Vegas, held January 8 to 11. The Consumer Electronics Show is huge, it covers the electronics industry from video to television to electronic games to computers. Computers have been a very small part of the show since the bust of the home computer market several years ago.

I will be there, covering the Amiga hardware and software in the Commodore booth. Commodore is expected to show the PC-10 and PC-10 IBM compatible computers, newly introduced to American markets. Some say the PC-40 will be shown, an IBM AT compatible machine.

Rumors say the Amiga 500 will be shown behind closed doors, to select dealers. At last word, I heard a well-known Amiga programmer was travelling to Germany to help finish the software for the Amiga 2500, so the chances are small that this machine will be shown at CES.

Spring WCCA
The Spring West Coast Commodore Association show will be held February 20 through 22 in San Francisco. For more details, call (800) 722-7927, or in California, (800) 722-7927. Judging by the fall WCCA show, this should be an Amiga-dominated event. Aside from the booths of Commodore products, there were worthwhile technical sessions. See you there!

The AMICUS & Fred Fish Public Domain Software Library

This software is collected from user groups and electronic bulletin boards around the nation. Each Amicus disk is nearly full, and is fully accessible from the Workbench. If source code is provided for any program, then the executable version is also present. This means that you don't need the C compiler to run these programs. An exception is granted for those programs only of use to people who own a C compiler.

The Fred Fish disk are collected by Mr. Fred Fish, a good and active friend of the Amiga.

Note: Each description line below may include something like 'S-O-E-D', which stands for 'source, object file, executable and documentation'. Any combination of these letters indicates what forms of the program are present. Basic programs are presented entirely in source code format.

AMICUS Disk 1

ABasic programs: Graphics

3DSolids	3d solids modeling program w/sample data files
Blocks	draws blocks
Cubes	draws cubes
Durer	draws pictures in the style of Durer
FScape	draws fractal landscapes
Hidden	3D drawing program, w/ hidden line removal
JPad	simple paint program
Optical	draw several optical illusions
PaintBox	simple paint program
Shuttle	draws the Shuttle in 3d wireframe
SpaceArt	graphics demo
Speaker	speech utility
Sphere	draws spheres
Spiral	draws color spirals
ThreeDee	3d function plots
Topography	artificial topography
Wheels	draws circle graphics
Xenos	draws fractal planet landscapes

ABasic programs: Tools

AddressBook	simple database program for addresses
CardFile	simple card file database program
Demo	multiwindow demo
KeyCodes	shows keycodes for a key you press
Menu	run many ABasic programs from a menu
MoreColors	way to get more colors on the screen at once, using aliasing
shapes	simple color shape designer Speakit speech and narrator demo

ABasic programs: Games

BrickOut	classic computer brick wall game
Othello	also known as 'go'
Saucer	simple shoot-em-up game
Spelling	simple talking spelling game
ToyBox	selectable graphics demo

ABasic programs: Sounds

Entertainer	plays that tune
HAL9000	pretends it's a real computer
Police	simple police siren sound
SugarPlum	plays "The Dance of the Sugarplum Fairies"

C programs:

ATerm	simple terminal program, S-E
cc	aid to compiling with Lattice C
decvnt	opposite of CONVERT for cross developers
Dotty	source code to the 'dotty window demo
echox	unix-style filename expansion, partial S, O-D
fasterfp	explains use of fast-floating point math
FixDate	fixes future dates on all files on a disk, S-E
freedraw	simple Workbench drawing program,S-E
GfxMem	graphic memory usage indicator, S-E
Grep	searches for a given string in a file, with documentation
ham	shows off the hold-and-modify method of color generation
IBM2Amiga	fast parallel cable transfers between an IBM and an Amiga
Mandel	Mandelbrot set program, S-E
moire	patterned graphic demo, S-E
objfix	makes Lattice C object file symbols visible to Wack, S-E
quick	quick sort strings routine
raw	example sample window I/O
setlace	turns on interlace mode, S-E
sparks	qix-type graphic demo, S-E

Other executable programs:

SpeechToy	speech demonstration
WhichFont	displays all available fonts

Texts:

68020	describes 68020 speedup board from CSA
Aliases	explains uses of the ASSIGN command
Bugs	known bug list in Lattice C 3.02
CLICard	reference card for AmigaDOS CLI
CLICommands	guide to using the CLI
Commands	shorter guide to AmigaDOS CLI commands
EdCommands	guide to the ED editor
Filenames	AmigaDOS filename wildcard conventions
HalfBright	explains rare graphics chips that can do more colors
ModemPins	description of the serial port pinout
RAMdisks	tips on setting up your RAM: disk
ROMWack	tips on using ROMWack
Sounds	explanation of the Instrument demo sound file format
Speed	refutation of the Amiga's CPU and custom chip speed
WackCmds	tips on using Wack

AMICUS Disk 2

C programs:

alib	AmigaDOS object library manager , S-E
ar	text file archive program, S-E
fixobj	auto-chops executable files
shell	simple CLI shell, S-E
sq, usq	file compression programs, S-E
YachtC	a familiar game, S-E
Make	a simple 'make' programming utility, S-E
Emacs	an early version of the Amiga text editor, S-E-D

Assembler programs:

bsearch.asm	binary search code
qsort.asm	Unix compatible qsort() function, source and C test program
setjmp.asm	setjmp() code for Lattice 3.02
SVprintf	Unix system V compatible printf()
trees.o	Unix compatible tree() function, O-D

(This disk formerly had IFF specification files and examples. Since this spec is constantly updated, the IFF spec files have been moved to their own disk in the AMICUS collection. They are not here.)

John Draper Amiga Tutorials:

Animate	describes animation algorithms
Gadgets	tutorial on gadgets
Menus	learn about Intuition menus

AMICUS Disk 3

C programs:

Xref	a C cross-reference gen., S-E
6bitcolor	extra-half-bright chip gfx demo, S-E
Chop	truncate (chop) files down to size, S-E
Cleanup	removes strange characters from text files
CR2LF	converts carriage returns to line feeds in Amiga files, S-E
Error	adds compile errors to a C file, S
Hello	window ex. from the RKM, S
Kermit	generic Kermit implementation, flakey, no terminal mode, S-E
Scales	sound demo plays scales, S-E
SkewB	Rubik cube demo in hi-res colors, S-E

AmigaBasicProgs(dir)

Automata	cellular automata simulation
CrazyEights	card game
Graph	function graphing programs
WitchingHour	a game

ABasiC programs:

Casino	games of poker, blackjack, dice, and craps
Gomoku	also known as 'othello'
Sabotage	sort of an adventure game

Executable programs:

Disassem	a 68000 disassembler, E-D
DpSlide	shows a given set of IFF pictures, E-D
Arrange	a text formatting program, E-D

Assembler programs:

Argoterm	a terminal program with speech and Xmodem, S-E

AMICUS Disk 4 **Files from the original Amiga Technical BBS**

Note that some of these files are old, and refer to older versions of the operating system. These files came from the Sun system that served as Amiga technical support HQ for most of 1985. These files do not carry a warranty, and are for educational purposes only. Of course, that's not to say they don't work.

Complete and nearly up-to-date C source to 'image.ed', an early version of the Icon Editor. This is a little flaky, but compiles and runs.

An Intuition demo, in full C source, including files: demomenu.c, demomenu2.c, demoreq.c, getascii.c, idemo.c, idemo.guide, idemo.make, idemoall.h, nodos.c, and txwrite.c

addmem.c	add external memory to the system
bobtest.c	example of BOB use
consoleIO.c	console IO example
creaport.c	create and delete ports
creastdi.c	create standard I/O requests
creatask.c	creating task examples
diskio.c	example of track read and write
dotty.c	source to the 'dotty window' demo
dualplay.c	dual playfield example
flood.c	flood fill example
freemap.c	old version of 'freemap'
geltools.c	tools for VSprites and BOBs
gfxmem.c	graphic memory usage indicator
hello.c	window example from RKM
inputdev.c	adding an input handler to the input stream
joystik.c	reading the joystick
keybd.c	direct keyboard reading
layertes.c	layers examples
mousport.c	test mouse port
ownlib.c, ownlib.asm	example of making your own library with Lattice
paratest.c	tests parallel port commands
seritest.c	tests serial port commands
serisamp.c	example of serial port use
prinintr.c	sample printer interface code
prtbase.h	printer device definitions
regintes.c	region test program
setlace.c	source to interlace on/off program
setparallel.c	set the attributes of the parallel port
SetSerial.c	set the attributes (parity, data bits) of the serial port
singplay.c	single playfield example
speechtoy.c	source to narrator and phonetics demo
timedely.c	simple timer demo
timer.c	exec support timer functions
timrstuf.c	more exec support timer functions
WhichFont.c	loads and displays all available system fonts

process.i and prtbase.i assmebler include files:

autorqstr.txt	warnings of deadlocks with autorequesters
consoleIO.txt	copy of the RKM console I/O chapter
diskfont.txt	warning of disk font loading bug
fullfunc.txt	list of #defines, macros, functions
inputdev.txt	preliminary copy of the input device chapter

License information on Workbench distribution license printer pre-release copy of the chapter on printer drivers, from RKM 1.1 v11fd.txt 'diff' of .fd file changes from version 1.0 to 1.1 v28v1.diff 'diff' of include file changes from version 28 to 1.0

AMICUS Disk 5 Files from the Amiga Link / Amiga Information Network

Note that some of these files are old, and refer to older versions of the operating system. These files are from Amiga Link. For a time, Commodore supported Amiga Link, aka AIN, for online developer technical support. It was only up and running for several weeks. These files do not carry a warranty, and are for educational purposes only. Of course, that's not to say they don't work.

A demo of Intuition menus called 'menudemo', in C source

whereis.c	find a file searching all subdirectories
bobtest.c	BOB programming example
sweep.c	sound synthesis example

Assembler files:

mydev.asm	sample device driver
mylib.asm	sample library example
mylib.i	
mydev.i	
asmsupp.i	
macros.i	assembler include files

Texts:

amigatricks	tips on CLI commands
extdisk	external disk specification
gameport	game port spec
parallel	parallel port spec
serial	serial port spec
v1.1update	list of new features in version 1.1
v1.1h.txt	'diff' of include file changes from version 1.0 to 1.1

Files for building your own printer drivers, including dospecial.c, epsondata.c, init.asm, printer.c, printer.link, printertag.asm, render.c, and wait.asm. This disk does contain a number of files describing the IFF specification. These are not the latest and greatest files, but remain here for historical purposes. They include text files and C source examples. The latest IFF spec is elsewhere in this library.

AMICUS Disk 6 IFF Pictures

This disk includes the DPSlide program, which can view a given series of IFF pictures, and the 'showpic' program, which can view each file at the click of an icon, and the 'saveilbm' program, to turn any screen into an IFF picture. The pictures include a screen from ArticFox, a Degas dancer, the guys at Electronic Arts, a gorilla, horses, King Tut, a lighthouse, a screen from Marble Madness, the Bugs Bunny Martian, a still from an old movie, the Dire Straits moving company, a screen from Pinball Contruction Set, a TV newcaster, the PaintCan, a world map, a Porsche, a shuttle mission patch, a tyrannosaurus rex, a planet view, a VISA card, and a ten-speed.

AMICUS Disk 7 DigiView HAM demo picture disk

This disk has pictures from the DigiView hold-and-modify video digitizer. It includes the ladies with pencils and lollypops, the young girl, the bulldozer, the horse and buggy, the Byte cover, the dictionary page, the robot and Robert. This includes a program to view each picture separately, and all together as separate, slidable screens.

AMICUS Disk 8

C programs:

Browse	view text files on a disk, using menus S-E-D
Crunch	removes comments and white space from C files, S-E
IconExec	EXECUTE a series of commands from Workbench S-E
PDScreen Dump	dumps Rastport of highest screen to printer
SetAlternate	sets a second image for an icon, when clicked once S-E
SetWindow	makes windows for a CLI program to run under Workbench S-E
SmallClock	a small digital clock that sits in a window menu bar
Scrimper	the screen printer in the fourth Amazing Computing, S-E

Amiga Basic Programs:

(Note: Many of these programs are present on AMICUS Disk 1. Several of these were converted to Amiga Basic, and are included here.)

AddressBook	a simple address book database
Ball	draws a ball
Cload	program to convert Compuserve hex files to binary, S-D
Clue	the game, Intuition driven
ColorArt	art drawing program
DeluxeDraw	the drawing program in the 3rd issue of Amazing Computing, S-D
Eliza	conversational computer psychologist
Othello	the game, as known as 'go'
RatMaze	3D ratmaze game
ROR	boggling graphics demo
Shuttle	draws 3D pictures of the space shuttle
Spelling	simple spelling program
YoYo	wierd zero-gravity yo-yo demo, tracks yo-yo to the mouse

Executable programs:

3Dcube	Modula-2 demo of a rotating cube
AltIcon	sets a second icon image, displayed when the icon is clicked
AmigaSpell	a slow but simple spelling checker, E-D
arc	the ARC file compression program, must-have for telecom, E-D
Bertrand	graphics demo
disksalvage	a program to rescue trashed disks, E-D
KwikCopy	a quick but nasty disk copy program: ignores errors, E-D
LibDir	lists hunks in an object file E-D
SaveILBM	saves any screen as an IFF picture E-D ??
ScreenDump	shareware screen dump program, E only
StarTerm	version 2.0, term program, Xmodem E-D

Texts:

LatticeMain	tips on fixing _main.c in Lattice
GDiskDrive	make your own 5 1/4 drive
GuruMed	explains the Guru numbers
Lat3.03bugs	bug list of Lattice C version 3.03
MForgeRev	user's view of the MicroForge hard drive
PrintSpooler	EXECUTE-based print spooling program

.BMAP files:

These are the necessary links between Amiga Basic and the system libraries. To take advantage of the Amiga's capabilities in Basic, you need these files. BMAPs are included for 'clist', 'console', 'diskfont', 'exec', 'icon', 'intuition', 'layers', 'mathffp', mathieeedoubas', 'mathieeesingbas', 'mathtrans', 'potgo', 'timer' and 'translator'.

AMICUS Disk 9

Amiga Basic Programs:

FlightSim	simple flight simulator program
HuePalette	explains Hue, Saturation, and Intensity
Requester	ex. of doing requesters from Amiga Basic
ScrollDemo	demonstrates scrolling capabilities
Synthesizer	sound program
WorldMap	draws a map of the world

Executable programs:

BoingI	latest BoingI demo,with selectable speed, E
Brush2C	converts an IFF brush to C data instructions, initialization code, E
Brush2Icon	converts IFF brush to an icon, E
Dazzle	graphics demo, tracks to mouse, E
DeclGEL	assembler program for stopping 68010 errors, S-E-D
Klock	menu-bar clock and date display, E
life	the game of life, E
TimeSet	Intuition-based way to set the time and date,
EMEmacs	another Emacs, more oriented to word processing, S-E-D
MyCLI	a CLI shell, works without the Workbench, S-E-D

Texts:

FnctnKeys	explains how to read function keys from Amiga Basic
HackerSin	explains how to win the game 'hacker'
Ist68010	guide to installing a 68010 in your Amiga
PrinterTip	tips on sending escape sequences to your printer
StartupTip	tips on setting up your startup-sequence file
XfrmrReview	list of programs that work with the Transformer

Printer Drivers:

Printer drivers for the Canon PJ-1080A, the C Itoh Prowriter, an improved Epson driver that eliminates streaking, the Epson LQ-800, the Gemini Star-10, the NEC 8025A, the Okidata ML-92, the Panasonic KX-P10xx family, and the Smith-Corona D300, with a document describing the installation process.

AMICUS Disk 10 Instrument sound demos

This is an icon-driven demo, circulated to many dealers. It includes the sounds of an acoustic guitar, an alarm, a banjo, a bass guitar, a boink, a calliope, a car horn, claves, water drip, electric guitar, a flute, a harp arpegio, a kickdrum, a marimba, a organ minor chord, people talking, pigs, a pipe organ, a Rhodes piano, a saxophone, a sitar, a snare drum, a steel drum, bells, a vibrophone, a violin, a wailing guitar, a horse whinny, and a whistle.

AMICUS Disk 11

C programs

dirutil	Intuition-based, CLI replacement file manager, S-E
cpri	shows and adjusts priority of CLI processes, S-E
ps	shows info about CLI processes, S-E
vidtex	displays Compuserve RLE pictures, S-E

AmigaBasic programs

pointered	pointer and sprite editor program
optimize	optimization ex ample from AC article
calendar	large, animated calendar, diary and date book program
amortize	loan amortizations
brushtoBOB	converts small IFF brushes to AmigaBasic BOB OBJECTS
grids	draw and play waveforms
hilbert	draws Hilbert curves
madlib	mad lib story generator
mailtalk	talking mailing list program
meadows3D	3D graphics program, from Amazing Computing™ article
mousetrack	mouse tracking example in hires mode
slot	slot machine game
tictactoe	the game
switch	pachinko-like game
weird	makes strange sounds

Executable programs

cp	unix-like copy command, E
cls	screen clear, S-E
diff	unix-like stream editor uses 'diff' output to fix files
pm	chart recorder performances indicator

Assembler programs

cls	screen clear and CLI arguments example

Modula-2

trails	moving-worm graphics demo
caseconvert	converts Modula-2 keywords to uppercase
Forth	Breshehan circle algorithm example
Analyze	12 templates for the spreadsheet Analyze!

There are four programs here that read Commodore 64 picture files. They can translate Koala Pad, Doodle, Print Shop and News Room graphics to IFF format. Of course, getting the files from your C-64 to your Amiga is the hard part.

AMICUS Disk 12

Executable programs

blink	'alink' compatible linker, but faster, E-D
clean	spins the disk for use with disk cleaners, E-D
epsonset	sends Epson settings to PAR: from menu, E-D
showbig	view hi-res pictures in low-res superbitmap, E-D
speaktime	tell the time, E-D
undelete	undeletes a file, E-D
cnvapldhm	converts Apple][low, medium and high res pictures to IFF, E-D
menued	menu editor produces C code for menus, E-D
quick	quick disk-to-disk nibble copier, E-D
quickEA	copies Electronic Arts disks, removes protection, E-D
txed 1.3	demo of text editor from Microsmiths, E-D

C programs

spin3	rotating blocks graphics demo, S-E-D
popcli	start a new CLI at the press of a button,
like	Sidekick, S-E-D
vsprite	VSprite example code from Commodore, S-E-D
AmigaBBS	Amiga Basic bulletin board program, S-D

Assembler programs

star10	makes star fields like Star Trek intro,S-E-D
Pictures	
Mount Mandelbrot	
	3D view of Mandelbrot set
Star Destroyer	hi-res Star Wars starship
Robot	robot arm grabbing a cylinder
Texts	
vendors	list of Amiga vendors, names, addresses
cardco	fixes to early Cardco memory boards
cinclude	cross-reference to C include files, who includes what
mindwalker	clues to playing the game well
slideshow	make your own slideshows from the Kaleidoscope disk

AMICUS Disk 13

Amiga Basic programs

Routines from Carolyn Scheppner of CBM Tech Support, to read and display IFF pictures from Amiga Basic. With documentation. Also included is a program to do screen prints in Amiga Basic, and the newest BMAP files, with a corrected ConvertFD program. With example pictures, and the SaveILBM screen capture program.

Routines to load and play FutureSound and IFF sound files from Amiga Basic, by John Foust for Applied Visions. With documentation and C and assembler source for writing your own libraries, and interfacing C to assembler in libraries. With example sound.

Executable programs

gravity	Sci Amer Jan 86 gravitation graphic simulation, S-E-D
Texts	
MIDI	make your own MIDI instrument interface, with documentation and a hi-res schematic picture.

AMICUS Disk 14

Several programs from Amazing Computing issues:

Tools

Dan Kary's	C structure index program, S-E-D

Amiga Basic programs

BMAP Reader	by Tim Jones
IFFBrush2BOB	by Mike Swinger
AutoRequester	example
DOSHelper	Windowed help system for CLI commands, S-E-D
PETrans	translates PET ASCII files to ASCII files, S-E-D
C Squared	Graphics program from Scientific American, Sept 86, S-E-D
crlf	adds or removes carriage returns from files, S-E-D
dpdecode	decrypts Deluxe Paint, removes copy protection, E-D
queryWB	asks Yes or No from the user, returns exit code, S-E
vc	VisiCalc type spreadsheet, no mouse control, E-D
view	views text files with window and slider gadget, E-D

Oing, Sproing, yaBoing, Zoing are sprite-based Boing! style demos, S-E-D

CLIClock, sClock, wClock are window border clocks, S-E-D

Texts

An article on long-persistance phospor monitors, tips on making brushes of odd shapes in Deluxe Paint, and recommendations on icon interfaces from Commodore-Amiga.

AMICUS 15

The C programs include:

'pr'	a file printing utility, which can print files in the background, and with line numbers and control character filtering.
'frm'	displays a chart of the blocks allocated on a disk.
'Ask'	questions an 'execute' file, returns an error code to control the execution in that batch file
'Stat'	an enhanced version of AmigaDOS 'status' command.
'Dissolve'	random-dot dissolve demo displays IFF picture slowly, dot by dot, in a random fashion.
'PopCLI2'	invoke new CLI window at the press of a key.

The executable programs include:

'Form'	file formatting program through the printer driver to select print styles
'DiskCat'	catalogs disks, maintains, sorts,merges lists of disk files
'PSound'	SunRize Industries' sampled sound editor & recorder
'Iconmaker'	makes icons for most programs
'Fractals'	draws great fractal seascapes and mountainscapes.
'3D Breakout'	3D glasses, create breakout in a new dimension
'AmigaMonitor'	displays lists of open files, memory use, tasks, devices and ports in use.
'Cosmoroids'	version of the 'asteroids' for the Amiga.
'Sizzlers'	high resolution graphics demo written in Modula 2.

Texts:

'ansi.txt'	explains escape sequences the CON: device responds to.
'FKey'	includes template for making paper to sit in the tray at the top of the Amiga keyboard.
'Spawn'	programmer's document from Commodore Amiga, describs ways to use the Amiga's multitasking capabilities in your own programs.

AmigaBasic programs:

'Grids'	draw sound waveforms, and hear them played.
'Light'	a version of the Tron light-cycle video game.
'MigaSol'	a game of solitaire.
'Stats'	program to calculate batting averages
'Money'	"try to grab all the bags of money that you can."

AMICUS 15 also includes two beautiful IFF pictures, of the enemy walkers from the ice planet in Star Wars, and a picture of a cheetah.

AMICUS 16

'juggler'	demo by Eric Graham. a robot juggler bouncing three mirrored balls, with sound effects. Twenty-four frames of HAM animation are flipped quickly to produce this image. You control the speed of the juggling. The author's documentation hints that this program might someday be available as a product.

IFF pictures

parodies of the covers of Amiga World and Amazing Computing magazines.	

C programs:

'Inputhandler'	example of making an input handler.
'FileZap3'	binary file editing program
'ShowPrint'	displays IFF picture, and prints it.
'Gen'	program indexes and retrieves C structures and variables declared in the Amiga include file system.

Executable Programs:

'FixHunk2'	repairs an executable program file for expanded memory
'ms2smus'	converts Music Studio files to IFF standard 'SMUS' format. I have heard this program might have a few bugs, especially in regards to very long songs, but it works in most cases.
'Missile'	Amiga version of the 'Missile Command' video game,

This disk also contains several files of scenarios for Amiga Flight Simulator II. By putting one of these seven files on a blank disk, and inserting it in the drive after performing a special command in this game, a number of interesting locations are preset into the Flight Simulator program. For example, one scenario places your plane on Alcatraz, while another puts you in Central Park.

Fred Fish Public Domain Software

Fred Fish Disk 1:

amigademo	Graphical benchmark for comparing amigas.
amigaterm	simple communications program with Xmodem
balls	simulation of the "kinetic thingy" with balls on strings
colorful	Shows off use of hold-and-modify mode.
dhrystone	Dhrystone benchmark program.
dotty	Source to the "dotty window" demo on the Workbench disk.
freedraw	A small "paint" type program with lines, boxes, etc.
gad	John Draper's Gadget tutorial program
gfxmem	Graphical memory usage display program
halfbrite	demonstrates "Extra-Half-Brite" mode, if you have it
hello	simple window demo
lattfp	accessing the Motorola Fast Floating Point l ibrary from C
palette	Sample program for designing color palettes.
trackdisk	Demonstrates use of the trackdisk driver.
requesters	John Draper's requester tutorial and example program.
speech	Sample speech demo program. Stripped down "speechtoy".
speechtoy	Another speech demo program.

Fred Fish Disk 2:

alib	Object module librarian.
cc	Unix-like frontend for Lattice C compiler.
dbug	Macro based C debugging package. Machine independent.
make	Subset of Unix make command.
make2	Another make subset command.
microemacs	Small version of emacs editor, with macros, no extensions
portar	Portable file archiver.
xrf	DECUS C cross reference utility.

Fred Fish Disk 3:

gothic	Gothic font banner printer.
roff	A "roff" type text formatter.
ff	A very fast text formatter
cforth	A highly portable forth implementation. Lots of goodies.
xlisp	Xlisp 1.4, not working correctly.

Fred Fish Disk 4:

banner	Prints horizontal banner
bgrep	A Boyer-Moore grep-like utility
bison	CNU Unix replacement 'yacc', not working.
bm	Another Boyer-Moore grep-like utility
grep	DECUS grep
kermit	simple portable Kermit with no connect mode.
MyCLI	Replacement CLI for the Amiga. Version 1.0
mandel	A Mandelbrot set program, by Robert French and RJ Mical

Fred Fish Disk 5:

cons	Console device demo program with supporting macro routines.
freemap	Creates a visual diagram of free memory
input.dev	sample input handler, traps key or mouse events
joystick	Shows how to set up the gameport device as a joystick.
keyboard	demonstrates direct communications with the keyboard.
layers	Shows use of the layers library
mandelbrot	IFF Mandelbrot program
mouse	hooks up mouse to right joystick port
one.window	console window demo
parallel	Demonstrates access to the parallel port.
printer	opening and using the printer, does a screen dump, not working
print.support	Printer support routines, not working.
proctest	sample process creation code, not working
region	demos split drawing regions
samplefont	sample font with info on creating your own
serial	Demos the serial port
singlePlayfield	Creates 320 x 200 playfield
speechtoy	latest version of cute speech demo
speech.demo	simplified version of speechtoy, with IO requests
text.demo	displays available fonts
timer	demos timer.device use
trackdisk	demos trakcdisk driver

Fred Fish Disk 6:

compress	like Unix compress, a file squeezer
dadc	analog clock impersonator
microemacs	upgraded version of microemacs from disk 2
mult	removes multiple occuring lines in files
scales	demos using sound and audio functions
setparallel	Allows changing parallel port parameters
setserial	Allows changing serial port parameters.
sortc	quicksort based sort program, in C
stripc	Strips comments and extra whitespace from C source

Fred Fish Disk 7:

This disk contains the executables of the game Hack, version 1.0.1.

Fred Fish Disk 8:

This disk contains the C source to Hack on disk 7.

Fred Fish Disk 9:

moire	Draws moire patterns in black and white
MVP-FORTH	Mountain View Press Forth, version 1.00.03A. A shareware version of FORTH from Fantasia Systems.

proff	a more powerful text formatting program
setlace	Program to toggle interlace mode on and off.
skewb	a rubic's cube type demo
sparks	moving snake Graphics demo

Fred Fish Disk 10:

conquest	An interstellar adventure simulation game
dehex	convert a hex file to binary
filezap	Patch program for any type of file.
fixobj	Strip garbage off Xmodem transferred files.
iff	Routines to read and write iff format files.
ld	simple directory program
ls	Minimal UNIX ls, with Unix-style wildcarding, in C
sq,usq	file squeeze and unsqueeze
trek73	Star Trek game
yachtc	Dice game.

Fred Fish Disk 11:

dpslide	slide show program for displaying IFF images with miscellaneous pictures

Fred Fish Disk 12:

amiga3d	Shows a rotating 3 dimensional solid "Amiga sign".
ArgoTerm	a terminal emulator program, written in assembler
arrow3d	Shows a rotating 3 dimensional wire frame arrow.
ld4	directory listing program
IconExec	
SetWindow	two programs for launching programs from Workbench that presently only work under CLI.
SetAlternate	Makes an icon show a second image when clicked once
StarTerm	terminal emulator, with ASCII Xmodem, dialer, more.

Fred Fish Disk 13:

A Bundle of Basic programs, including:

Jpad	toybox	ezspeak	mandlebrot
xmodem	3dsolids	addbook	algebra
ror	amgseq1	amiga-copy	band
bounce	box	brickout	canvas
cardfi	circle	colorcircles	Copy
cubes1	cutpaste	date	dogstar
dragon	draw	dynamictriangle	
Eliza	ezterm	fillibuster	fractal
fscape	gomoku	dart	haiku
hal9000	halley	hauntedM	hidden
join	loz	mandel	menu
minipaint	mouse	Orthello	patch
pena	pinwheel	gbox	random-circles
Readme	rgb	rgbtest	Rord
sabotage	salestalk	shades	shapes
shuttle	sketchpad		spaceart
speakspeach	speecheasy		spell
sphere	spiral	striper	superpad
suprshr	talk	terminal	termtest
tom	topography		triangle
wheels	xenos		xmostriper

(note: some programs are Abasic, most are Amigabasic, and some programs are presented in both languages)

Fred Fish Disk 14:

amiga3d	update of #12, includes C source to a full hidden surface removal and 3D graphics
beep	Source for a function that generates a beep sound
dex	extracts text from within C source files
dimensions	demonstrates N dimensional graphics
filezap	update of disk 10, a file patch utility
gfxmem	update of disk 1, graphic memory usage indicator
gi	converts IFF brush files to Image struct, in C text.
pdterm	simple ANSI VT100 terminal emulator, in 80 x 25 screen
shell	simple Unix 'csh' style shell
termcap	mostly Unix compatible 'termcap' implementation.

Fred Fish Disk 15:

Blobs	graphics demo, like Unix 'worms'
Clock	simple digital clock program for the title bar
Dazzle	An eight-fold symmetry dazzler program. Really pretty!
Fish	double buffered sequence cycle animation of a fish
Monopoly	A really nice monopoly game written in AbasiC.
OkidataDump	Okidata ML92 driver and WorkBench screen dump program.
Polydraw	A drawing program written in AbasiC.
Polyfractals	A fractal program written in AbasiC.

Fred Fish Disk 16:

A complete copy of the latest developer IFF disk

Fred Fish Disk 17:

The NewTek Digi-View video digitizer HAM demo disk

Fred Fish Disk 18:

AmigaDisplay	dumb terminal program with bell, selectable fonts
Ash	Prerelease C Shell-like shell program, history, loops, etc.
Browser	wanders a file tree, displays files, all with the mouse
MC68010	docs on upgrading your Amiga to use a 68010
Multidim	rotate an N dimensional cube with a joystick
PigLatin	SAY command that talks in Pig Latin
Scrimper	Screen image printer
Xlisp1.6	source, docs, and executable for a Lisp interpreter.

Fred Fish Disk 19:

BlackJack	text-oriented blackjack game
JayMinerSlides	Slides by Jay Miner, Amiga graphics chip designer, showing flowchart of the Amiga internals, in 640 x 400.
Keymap_Test	test program to test the keymapping routines
LockMon	Find unclosed file locks, for programs that don't clean up.

Fred Fish Disk 20:

AmigaToAtari	converts Amiga object code to Atari format
DiskSalv	program to recover files from a trashed AmigaDOS disk.
Hash	example of the AmigaDOS disk hashing function
Hd	Hex dump utility ala Computer Language magazine, April 86
MandelBrots	Mandelbrot contest winners
MultiTasking	Tutorial and examples for Exec level multitasking
Pack	strips whitespace from C source
PortHandler	sample Port-Handler program that performs. Shows BCPL environment clues.
Random	Random number generator in assembly, for C or assembler.
SetMouse2	sets mouse port to right or left port.
SpeechTerm	terminal emulator with speech capabilities, Xmodem
TxEd	Demo editor from Microsmiths Charlie Heath

Fred Fish Disk 21

This is a copy of Thomas Wilcox's Mandelbrot Set Explorer disk. Very good!

Fred Fish Disk 22

This disk contains two new "strains" of microemacs.

Lemacs	version 3.6 by Daniel Lawrence. For Unix V7, BSD 4.2, Amiga, MS-DOS, VMS. Uses Amiga function keys, status line, execute, startup files, more.
Pemacs	By Andy Poggio. New features include <ALT> keys as Meta keys, mouse support, higher priority, backup files, word wrap, function keys.

Fred Fish Disk 23

Disk of source for MicroEmacs, several versions for most popular operating systems on micros and mainframes. For people who want to port MicroEmacs to their favorite machine.

Fred Fish Disk 24:

Conques	interstaller adventure simulation game
Csh	update to shell on Disk 14, with built in commands,named variables, substitution.
Modula-2	A pre-release version of the single pass Modula-2 compiler originally developed for MacIntosh at ETHZ. This code was transmitted to the AMIGA and is executed on the AMIGA using a special loader. Binary only.

Fred Fish Disk 25

Graphic Hack	A graphic version of the game on disks 7 and 8

Fred Fish Disk 25

This is the graphics-oriented Hack game by John Toebes. Only the executable is present.

Fred Fish Disk 26

UnHunk	Processes the Amiga "hunk" loadfiles. Collect code, data, and bss hunks together, allows individual specificatio of code, data, and bss origins, and generates binary file with format reminiscent of Unix "a.out" format. The output file can be easily processed by a separate program to produce Motorola "S-records" suitable for downloading to PROM programmer. By Eric Black.
C-kermit	Port of the Kermit file transfer program and server.
Ps	Display and set process priorities
Archx	Yet another program for bundling up text files and mailing or posting them as a single file unit.

Fred Fish Disk 27

ABdemos	Amiga Basic demos from Carolyn Scheppner.
NewConvertFD	creates .bmaps from fd files.
BitPlanes	finds addresses of and writes to bitplanes of the screen's bitmap.
AboutBmaps	A tutorial on creation and use of bmaps.
LoadILBM	loads and displays IFF ILBM pics.
LoadACBM	loads and displays ACBM pics.
ScreenPrint	creates a demo screen and dumps it to a graphic printer.
Disassem	Simple 68000 disassembler. Reads standard Amiga object files and disassembles the code sections. Data sections are dumped in hex. The actual disassember routines are set up to be callable from a user program so instructions in memory can be disassembled dynamically. By Bill Rogers.
DvorakKeymap	Example of a keymap structure for the Dvorak keyboard layout. Untested but included because assembly examples are few and far between. By Robert Burns of C-A.
Hypocycloids	Spirograph, from Feb. 84 Byte.
LinesDemo	Example of proportional gadgets to scroll a SuperBitMap.
MemExpansion	Schematics and directions for building your own homebrew 1 Mb memory expansion, by Michael Fellinger.
SafeMalloc	Program to debug 'malloc()' calls
ScienceDemos	Convert Julian to solar and sidereal time, stellar positions and radial velocity epoch calculations and Galilean satellite plotter. By David Eagle.

Fred Fish Disk 28

ABasic games by David Addison: Backgammon , Cribbage, Milestone, and Othello

Cpp	DECUS 'cpp' C preprocessor, and a modified 'cc' that knows about the 'cpp', for Manx C.
Shar	Unix-compatible shell archiver, for packing files for travel.
SuperBitMap	Example of using a ScrollLayer, syncing SuperBitMaps for printing, and creating dummy RastPorts.

Fred Fish Disk 29

AegisDraw Demo	Demo program without save and no docs.
Animator Demo	Player for Aegis Animator files
Cc	Unix-like front-end for Manx C.
Enough	Tests for existance of system resources, files, devices.
Rubik	Animated Rubik's cube program
StringLib	Public domain Unix string library functions.
Vt100 .	VT-100 terminal emulator with Kermit and Xmodem protocols

Fred Fish Disk 30

Several shareware programs. The authors request a donation if you find their program useful, so they can write more software.

BBS	an Amiga Basic BBS by Ewan Grantham
FineArt	Amiga art
FontEditor	edit fonts, by Tim Robinson
MenuEditor	Create menus, save them as C source, by David Pehrson
StarTerm3.0	Very nice telecommunications by Jim Nangano

(Fred Fish Disk#30 is free when ordered with at least three other disks from the collection.)

Fred Fish Disk 31

Life	Life game, uses blitter to do 19.8 generations a second.
Mandelbrot	Version 3.0 of Robert French's program.
MxExample	Mutual exclusion gadget example.
RamSpeed	Measure relative RAM speed, chip and fast.
Set	Replacement for the Manx "set" command for environment variables, with improvements.
Tree	Draws a recursive tree, green leafy type, not files.
TxEd	Crippled demo version of Microsmith's text editor, TxEd.
VDraw	Full-featured drawing program by Stephen Vermeulen.
Xicon	Invokes CLI scripts from Icon
Ticon	Displays text files from an icon.

Fred Fish Disk 32

Address	Extended address book written in AmigaBasic.
Calendar	Calendar/diary program written in AmigaBasic.
DosPlus1	First volume of CLI oriented tools for developers.
DosPlus2	Second volume of CLI oriented tools for developers.
Executables only:	
MacView	Views MacPaint pictures in Amiga low or high res, no sample pictures, by Scott Evernden.
Puzzle	Simulation of puzzle with moving square tiles.
ShowHAM	View HAM pictures from CLI.
Solitaire	ABasiC games of Canfield and Klondike, from David Addison.
Spin3	Graphics demo of spinning cubes, double-buffered example.
Sword	Sword of Fallen Angel text adventure game written in Amiga Basic.
Trails	Leaves a trail behind mouse, in Modula-2

Fred Fish Disk 33

3dstars	3d version of the "stars" program below.
Bigmap	Low-level graphics example scrolls bitmap with ScrollVPort.
Dbuf.gels	Double-buffered animation example for BOBs and VSprites.
DiskMapper	Displays sector allocation of floppy disks.
MemView	View memory in real time, move with joystick.
Oing	Bouncing balls demo
Sproing	Oing, with sound effects.
ScreenDump	Dumps highest screen or window to the printer.
Sdb	Simple database program from a DECUS tape.
Stars	Star field demo, like Star Trek.
TermPlus	Terminal program with capture, library, function keys, Xmodem, CIS-B protocols.
Vt100	Version 2.0 of Dave Wecker's VT-100 emulator, with scripts and function keys.

Fred Fish Disk 34

Alint	Support files for Gimpel's 'lint' syntax checker
Blink	PD 'alink' compatible linker, faster, better.
Browser	Updated to FF 18 'browser', in Manx, with scroll bars, bug fixes.
Btree	b-tree data structure examples
Btree2	Another version of 'btree'
Calendar	Appointment calendar with alarm.
Less	File viewer, searching, position by percent, line number.
NewFonts	Set of 28 new Amiga fonts from Bill Fischer
Pr	Background print utility, style options, wildcards.
Requester	Deluxe Paint-type file requester, with sample.

Fred Fish Fred Fish Disk 35

ASendPacket	C example of making asynchronous I/O calls to a DOS handler, written by C-A
ConsoleWindow	C example of getting the Intuition pointer a CON: or RAW: window, for 1.2, by C-A.
DirUtil	Walk the directory tree, do CLI operations from menus
DirUtil2	Another variant of Dirutil.
FileRequester	Lattice C file requester module, with demo driver, from Charlie Heath.
MacView	Views MacPaint pictures in Amiga low or high res, with sample pictures, by Scott Evernden.
Plop	Simple IFF reader program
PopCLI	Sidekick-style program invokes a new CLI, with automatic screen blanking.
QuickCopy	Devenport disk copiers duplicate copy-protected disks.
ScrollPf	Dual playfield example, from C-A, shows 400 x 300 x 2 bit plane playfield on a 320 x 200 x 2 plane deep playfield.
SendPacket	General purpose subroutine to send AmigaDos packets.
SpriteMaker	Sprite editor, can save work as C data structure. Shareware by Ray Larson.
Tracker	Converts any disk into files, for electronic transmission. Preserves entire file structure. Shareware by Brad Wilson.
TriClops 3-D	space invasion game, formerly commercial, now public domain. From Geodesic Publications.
Tsize	Print total size of all files in subdirectories.
Unifdef	C preprocessor to remove given #ifdef'd sections of a file, leaving the rest alone. By Dave Yost.
Vttest	VT-100 emulation test program. Requires a Unix system.

Fred Fish Disk 36

Acp	Unix-like 'cp' copy program
Clock	Updated version of clock on disk 15.
Csh	Manx 'csh'-like CLI, history, variables, etc.
DietAid	Diet planning aid organizes recipes, calories
Echo	Improved 'echo' command with color, cursor addressing
FixHunk	Fixs programs to let them run in external memory.
Fm	Maps the sectors a file uses on the disk.
KickBench	Docs, program to make a single disk that works like a Kickstart and Workbench.
Lex	Computes Fog, Flesch, and Kincaid readability of text files.
TunnelVision	David Addison ABasic 3D maze perspective game.
Vc	Visicalc-like spreadsheet calculator program.
Vt100	Version 2.2 of Dave Wecker's telecom program
YaBoing	Oingl style game program shows sprite collision detects

Fred Fish Disk 37

This disk is a port of Timothy Budd's Little Smalltalk system, done by Bill Kinnersley at Washington State University.

Fred Fish Disk 38

CSquared	Sep 86 Sci American, Circle Squared algorithm
FixObj	Strips garbage off Xmodem transfered object files
Handler	AmigaDOS handler (device) example from C-A
Hp-10c	Mimics a HP-10C calculator, written in Modula-2
IFFEncode	Saves the screen as an IFF file
IffDump	Dumps info about an IFF file
Jsh	BDS C-like CLI shell
NewStat	STATUS-like program, shows priority, processes
Reversi	Game of Reversi, version 6.1
UUdecode	Translate binary files to text, Unix-like programs
Vdraw	Drawing program, version 1.14
VoiceFiler	DX MIDI synthesizer voice filer program
Window	Example of creating a DOS window on a custom screen

Fred Fish Disk 39

AnsiEcho	'echo', 'touch', 'list', 'cls' written in assembler.
Display	Displays HAM images from a ray-tracing program, with example pictures.
Driver	Example device driver source, acts like RAM: disk
Xlisp	XLisp 1.7, executable only

Fred Fish Disk 40

Ahost	Terminal emulator with Xmodem, Kermit and CIS B protocols, function keys, scripts, RLE graphics and conference mode.
AmigaMonitor	Dynamically displays the machine state, such as open files, active tasks, resources, device states, interrupts, libraries, ports, etc.
Arc	Popular file compression system, the standard for transitting files
AreaCode	Program that decodes area codes into state and locality.
Blink	'alink' replacement linker, version 6.5
Cosmo	An 'asteriods' clone.
Dg210	Data General D-210 Terminal emulator
DirUtil	Windowed DOS interface program, version 1.4
DOSHelper	Windowed AmigaDOS CLI help program
PagePrint	Prints text files with headers, page breaks, line numbers
PopCLI	Starts a new CLI with a single keystroke, from any program, With a screen-saver feature. Version 2, with source.
SpriteEd	Sprite Editor edits two sprites at a time
X-Spell	Spelling checker allows edits to files

FF 41

AmigaVenture AmigaBasic.	Create your own text adventure programs in
Csh Executable only	Version 2.03 of Dillon's Csh-like shell.
Dbug to FF #2	Macro based C debugging package, update
DualPlayField manual	example from CBM, update to Intuition
GetFile	Heath's file requester, with source
LatXref	Cross reference of Lattice 3.10 header files
Lines	Line drawing demo program
SetFont	Changes font used in a CLI window
Vt100	Version 2.3 of the VT-100 terminal program.

FF 42

This disk contains an Amiga version of MicroGNUEmacs.

FF 43

BasicBoing	AmigaBasic program demos page flipping of a 3D cube
Bbm	Demo copy of B.E.S.T. Business Management System.
BbsList	A list of Amiga Bulletin Board Systems
Cc	C compiler frontends for Manx and Lattice C
Copper	A hardware copper list disassembler
InstIFF	Converts Instruments demo sounds to IFF sampled sounds
PopColours	Adjust RGB colors of any screen
SpriteClock	Simple clock is displayed on a sprite above all screens
ST Emulator	Non-serious Atari ST emulator
WBrun	Lets Workbench programs be run from the CLI
Wild	Two Unix shell style wildcard matching routines

FF 44

Icons	Miscellaneous icons
NewIFF	New IFF material from CBM for sampled voice and music files
RayTracePics	The famous ray-tracing pictures, from FF #39, now converted to IFF HAM format for *much* faster viewing.
ViewILBM	Displays normal and HAM ILBM files

FF 45

Clue	Clue board game
Make	Another 'make', with more features
Pictures	Miscellaneous pictures
Update	Updates an older disk with newer files from another disk
WhereIs	Searches a disk for files of given name

FF 46

Asm	Shareware 68010 macro assembler, ROM Kernal Manual compatible
CheckModem	'execute' file program detects presence of modem
Egad	Gadget editor from the Programmers Network
Jive	Transforms a file from English to Jive.
My.lib	A binary only copy of Matt's alternate runtime library. Author: Matt Dillon
ProffMacros	Subset Berkeley 'ms' and 'mm' macros for 'proff'
ValSpeak	Transforms a file from English to Valley Speak.

In Conclusion

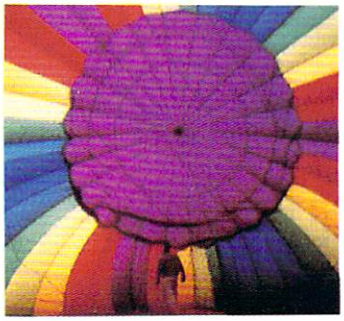
To the best of our knowledge, the materials in this library are freely distributable. This means they were either publicly posted and placed in the Public Domain by their Author, or they have restrictions published in their files to which we have adhered. If you become aware of any violation of the author's wishes, please contact us by mail.

•AC•

Index of Advertisers

Support the Amiga™ and Amazing Computing™, Write!

Your thoughts, experiences, and programs are needed by others. For an Author's guide, write to: Author's Guide, PiM Publications, Inc., P.O.Box 869, Fall River, MA. 02722.



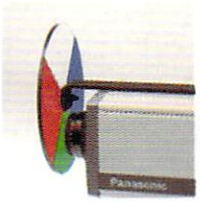

Actual unretouched photos

DIGI-VIEW

brings the world into your Amiga™!

With **Digi-View** and a video camera, your **Amiga** can see! Faces, logos, artwork . . . anything you can imagine! Simply point your camera and click the mouse. In seconds, whatever the camera sees is painlessly transformed into a computer image that can be printed, stored on disk, or transferred to other programs. Imagine how quickly and easily you can generate stunning video art and animation when you start with high quality digitized photographs or artwork.

Sophisticated software included with **Digi-View** makes it easy to produce dazzling, broadcast-quality color images. Intuitive, on-screen controls are as easy to use as the knobs on your T.V. set. **Digi-View** can capture images in several modes, including 320x200 pixels with up to 4096 colors on screen ("hold-and-modify" mode), and the incredibly detailed 640x400 high resolution mode.



*The key to **Digi-View's** incredible color resolution is this color separation filter which attaches to your black-and-white or color video camera.**

- IFF disk format works with Digi-Paint™, DeluxePaint™, DeluxeVideo™, DeluxePrint, Aegis Images™, Aegis Animator, and more!
- Saves time! No more hours of freehand drawing and redrawing.
- Send photos over the telephone with your modem and terminal software.
- Capture images for scientific image processing or pattern recognition.
- Spice up business graphics — slide show program included.
- Incorporate photos in posters and greeting cards.
- Use **Digi-View** pictures in your BASIC programs.
- Catalog images with IFF database programs.
- Make red/blue 3D photos.
- A powerful tool for commercial graphic artists!

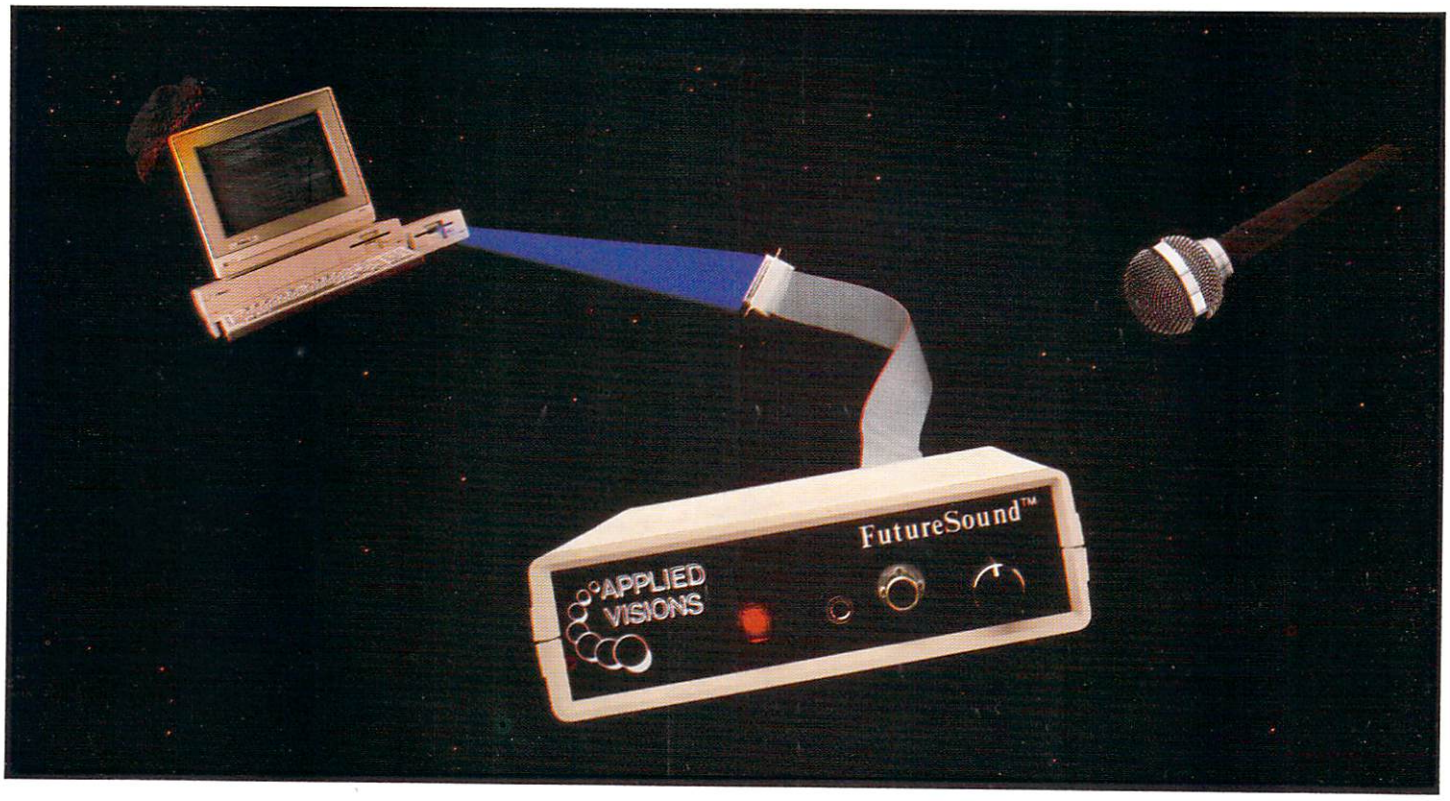
Panasonic WV-1410 video camera w/lens $280
CS-1L Copy stand w/lights . . . . . $ 75

Only $199.95
includes video digitizer module, color separation filter, software and manual.

Orders Only (800) 358-3079 ext. 342
Customer Service (913) 354-9332

NEWTEK
INCORPORATED
701 Jackson • Suite B3 • Topeka, KS • 66603

* Digi-View software version 2.0 (or newer) required to use color camera. For maximum resolution use monochrome camera with 2.1 interlace. High-res color modes require 1 Meg expansion RAM.



"Open the pod bay doors, HAL..."

Programmers cast their vote!

Right now, leading software developers are hard at work on the next generation of Amiga® products. To add the spectacular sound effects we've all come to expect from Amiga software, they are overwhelmingly choosing one sound recording package... FutureSound. As one developer put it, "FutureSound should be standard equipment for the Amiga."

FutureSound the clear winner...

Why has FutureSound become the clear choice for digital sound sampling on the Amiga? The reason is obvious: a hardware design that has left nothing out. FutureSound includes two input sources, each with its own amplifier, one for a microphone and one for direct recording; input volume control; high speed 8-bit parallel interface, complete with an additional printer port; extra filters that take care of everything from background hiss to interference from the monitor; and of course, a microphone so that you can begin recording immediately.

What about software?

FutureSound transforms your Amiga into a powerful, multi-track recording studio. Of course, this innovative software package provides you with all the basic recording features you expect. But with FutureSound, this is just the beginning. A forty-page manual will guide you through such features as variable sampling rates, visual editing, mixing, special effects generation, and more. A major software publisher is soon to release a simulation with an engine roar that will rattle your teeth. This incredible reverberation effect was designed with FutureSound's software.

APPLIED VISIONS

Question: What can a 300 pound space creature do with these sounds?

Answer: Anything he wants.

Since FutureSound is IFF compatible (actually three separate formats are supported) your sounds can be used by most Amiga sound applications. With FutureSound and Deluxe Video Construction Set from Electronic Arts, your video creations can use the voice of Mr. Spock, your mother-in-law, or a disturbed super computer.

Programming support is also provided. Whether you're a "C" programming wiz or a Sunday afternoon BASIC hacker, all the routines you need are on the non-copy protected diskette.

Your Amiga dealer should have FutureSound in stock. If not, just give us a call and for $175 (VISA, MasterCard or COD) we'll send one right out to you. Ahead warp factor one!

Applied Visions, Inc., Suite 2200, One Kendall Square
Cambridge, MA 02139 (617) 494-5417

LOGISTIX

IS

Spreadsheet+Timesheet+Database+Graphics+Project Management=

Successful Management!

Computer software has always been the limiting factor in your business operation. Now Progressive Peripherals & Software, Inc. has broken through this barrier to bring you a completely new concept in integrated business software. *Logistix*, the software that outweighs all other business software products!

Logistix is the planning tool designed to provide professional business people with a solution to the limitations of current spreadsheets and project management software. With time and project management built in to the same work area, you can now solve the simultaneous problems of time and money that effect your business for today and tomorrow.

Logistix is the first and only business software program to actually integrate both time and financial functions. Face it: Time is Money. With *Logistix*, you can schedule personnel, manage shipments, plan work schedules and production flows. This allows you to monitor, project, and protect your business interests better than ever before.

Logistix offers you the four dynamic features needed to widen the horizons of success for your business. *Logistix* combines together, in one worksheet area, a large and sophisticated spreadsheet, presentation quality color graphics, database functions and, of course, powerful and flexible time and project management. Best of all, it is all designed with the business person in mind.

Logistix has built-in sideways print functions, supports over 20 international currency symbols, and offers complete support documentation, including a examples diskette. *Logistix* reads 1-2-3 files and many other file formats, so no time is lost reformatting existing data. You'll be finding business solutions faster than ever before.

Logistix is the most modern and intelligent planning tool ever designed for business people, by business people. It's loaded with features you expect to find in programs costing hundreds of dollars more. When it comes to business software, *Logistix* helps you see the future possibilities of your business.

For more information about *Logistix* call or write us today.

AMIGA

IBM®

HEWLETT PACKARD

WHEN QUALITY COUNTS!

PROGRESSIVE PERIPHERALS & SOFTWARE

464 KALAMATH STREET
DENVER, COLORADO 80204
303-825-4144
TELEX: 888837

UNLEASH THE AWESOME POWER OF THE AMIGA!™

WITH PAL SYSTEMS

- Supports Three Half Height Devices
  - Hard Disks
  - Tape Backup
  - CD ROM
- Five DMA Expansion Slots
- Battery Backed Clock/Calendar
- Whisper Fan
- Auto-Configure
- 200 Watt Power Supply
- DMA Hard Disk Controller (ST506/412)
  - Optional additional SCSI
- 100% Zorro Compatible
- 1 to 9.5 Megabytes of Fast RAM

WITH PAL Jr

- One Megabyte of Fast RAM
- DMA Hard Disk Controller
- 20 Megabyte Hard Disk
- Auto-Configure
- DMA SCSI Pass-through for further expansion

Suggested retail price only $1495.

WITH INFOMINDER

The Information Manager. Hierarchial Database that allows you to organize and display text and graphical files, e.g. Real Estate Listings, Personnel Files, Digitized X-Rays, Geographical Maps, etc. Fully supports multi-tasking. Fast access by menu or outline. INFOMINDER will revolutionize the way you store and access both textual and graphical information. Get INFOMINDER today at the special introductory price of only $89.95.

WITH TIC

The TIC provides your Amiga with a tiny battery backed clock/calendar that conveniently plugs into the second joystick port. The TIC's 3-year battery will maintain time even if temporarily removed from the Amiga. Change the Amiga's internal time simply by moving the displayed clock's hands with the mouse. Set your Amiga's time once and for all. It's about time for TIC. Suggested retail price only $59.95.

BYTE by BYTE™ CORPORATION

Arboretum Plaza II
9442 Capital of Texas Highway
Suite 150
Austin, TX 78759
(512) 343-4357

AMIGA is a trademark of Commodore-Amiga, Inc.